1987-03

Further development of a one-dimensional unsteady Euler code for wave rotor applications

Johnston, David T.

Monterey, California: U.S. Naval Postgraduate School

http://hdl.handle.net/10945/22817

NPS67-87-002

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

FURTHER DEVELOPMENT OF A
ONE-DIMENSIONAL UNSTEADY EULER CODE
FOR WAVE ROTOR APPLICATIONS

by

David T. Johnston

March 1987

Thesis Advisor: Ray P. Shreeve

Approved for public release; distribution is unlimited

Prepared for: Naval Air Systems Command
Washington, DC 20361

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral Robert C. Austin
Superintendent

David A. Schrady
Provost

This thesis was prepared in conjunction with research sponsored in part by the Naval Air Systems Command, Washington, DC, under Air Task #A931931E/186A/7R024-03-001.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS67-87-002	5 MONITORING ORGANIZATION REPORT NUMBER(S) NPS67-87-002
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable) Code 67	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION Naval Air Systems Command; 8b OFFICE SYMBOL (If applicable) AIR 931E	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N0001987WR7R048
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
61153N	7R024	03	001

11 TITLE (Include Security Classification)
FURTHER DEVELOPMENT OF A ONE-DIMENSIONAL UNSTEADY EULER CODE FOR WAVE ROTOR APPLICATIONS

12 PERSONAL AUTHOR(S)
Johnston, David T.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____	1987, March	204

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	QAZ1D; Wave Rotor; Unsteady Euler Code; Computational Fluid Dynamics

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The EULER1 Fortran program for computing one-dimensional unsteady flow based on the QAZ1D method of Verhoff was extended to provide tracking and correcting of discontinuities, flow to right or left and open and closed end boundary conditions. The program was run on four shock tube test cases to verify accuracy and range of capability of the revised E1DV2 code. Further extensions and test verifications are recommended so that the code is suitable for wave rotor applications.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Raymond P. Shreeve	22b TELEPHONE (Include Area Code) (408) 646-2593; 22c OFFICE SYMBOL Code 67Sf

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited

Further Development of a One-Dimensional Unsteady Euler Code for Wave Rotor Applications

by

David T. Johnston
Lieutenant, United States Navy
B.S., Cornell University, 1979

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1987

ABSTRACT

The EULER1 Fortran program for computing one-dimensional unsteady flow based on the QAZ1D method of Verhoff was extended to provide tracking and correcting of discontinuities, flow to right or left and open and closed end boundary conditions. The program was run on four shock tube test cases to verify accuracy and range of capability of the revised E1DV2 code. Further extensions and test verifications are recommended so that the code is suitable for wave rotor applications.

Thesis
J...3
c.1

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

TABLE OF SYMBOLS

TEXT	E1DV2	DEFINITION
A	A	Speed of Sound
	*A	Denotes the value of a variable at the node to the left of a discontinuity, * can be any variable name
A_i/A_j	AR	The ratio of sound speed across a shock. A/B (shock moving right), B/A (shock moving left)
	*B	Denotes the value of a variable at the node to the right of a discontinuity
	BDRY	3 denotes left boundary, 2 the right boundary
	*BD	Variable at phantom node
	CNTACT	3 digit variable denoting contact surface location, direction of travel, and if it crosses a node
	COUNT	Counter for graphics routines
	CSDIR	Contact surface direction, 2 to the right, 3 to the left
	CSRMN	Riemann variable change across a contact surface
	D2	Density ratio at first node inside boundary
	DARRAY	Array of density for plotting
	DELAH	Change in A from I to I+1
	DELAL	Change in A from I-1 to I
	DELQQH	Change in QQ from I to I+1
	DELQQL	Change in QQ from I-1 to I
	DELRRH	Change in RR from I to I+1

	DELRRL	Change in RR from I-1 to I
	DELSH	Change in S from I to I+1
	DELSL	Change in S from I-1 to I
δt	DELT	Time step
δt_{ex}	DELTEX	The excess time in a time step when the shock is exactly at the solid wall
∂t_{wl}	DELTWL	The time for the shock to reach the wall
	DELQH	Change in Q from I to I+1
	DELQL	Change in Q from I-1 to I
Δs	DELX	Interpolation distance (LMD+DELT)
ρ	DENS	Density
	DLCD	Density to the left of the contact discontinuity in the exact solution
	DLSH	Density to the left of the shock in the exact solution
	DLTA**	Prefix which indicates the spatial change in ** for one time step
	DQ	The jump in velocity across the shock divided by the sound speed at B (right) or A (left)
	DR	The ratio of the density across a shock, B/A (right), B/A (left)
	DRI	Initial density ratio across the diaphragm
	EE	Desired precision for characteristic calculations
	E(K)	Actual error in characteristic slope calculation
	EREIMN	The jump in QQ across the shock calculated analytically as a function of w
γ	G	Gamma (ratio of specific heats)

GRAPHS	For graphical output, 0 = none (tabular), 1 = plots all variables, 2 = compares density with exact solution
G1	1/(G-1)
G2	2/(G-1)
H	1/(N-1)
HALT	Terminates program if 1, set by conditions not coded
INTEG(K)	Result of integrating Z(K)
**INT	Value of ** interpolated between nodes on the current time level
I2	Number of the node to the right of a discontinuity
JSTOP	Number of time levels to be calculated
LBDDR	Left boundary density ratio
LBDDRI	Left boundary density ratio at time zero
LBDPR	Left boundary pressure ratio
LBDPRI	Left boundary pressure ratio at time zero
LBDPRS	Value of 0 denotes constant pressure at left boundary, 1 denotes adjustable pressure at the left boundary
LBDTR	Left boundary temperature ratio
LBDTRI	Left boundary temperature ratio at time zero
LBNDRY	Denotes left boundary condition, open or closed
λ LMD(K)	The characteristic trajectories in the space-time plane (q+A , q-A , q)
LNODE	Array of left most node to be corrected in CORRCT
LWPRES	Denotes which side of diaphragm has low pressure

	LXX	Node defining the left interval
	MREIMN	The measured jump in QQ across the shock, from A to B
	N	Number of Spacial Nodes (odd number)
	ND	Double precision value of N
	NEW**(I)	Stored values of ** for the next time level
	PARRAY	Array of pressures for plotting
P_i/P_j	PR	The ratio of the pressure across a shock, A/B (right), B/A (left)
P	PRESS	Pressure
	PRI	Initial pressure ratio across the diaphragm
	PLTCNT	Counter for graphics routines
$\frac{\partial *}{\partial s}$	*PRIM(K)	Suffix which indicates the spatial derivative of * at the current time level
	P2	Pressure ratio at first node inside boundary
q	Q	Absolute fluid velocity
	QARRAY	Array of velocities for plotting
	QLBD	Initial velocity at left boundary
	QLI	Initial velocity left of the diaphragm
	QRBD	Initial velocity at right boundary
	QRI	Initial velocity right of the diaphragm
Q	QQ	Q+A*S (extended Riemann variable)
	RBDDR	Right boundary density ratio
	RBDDRI	Initial right boundary density ratio
	RBDPR	Right boundary pressure ratio
	RBDPRI	Initial right boundary pressure ratio

	RBDPRS	Value of 0 denotes a constant pressure at right boundary, while 1 is for adjustable pressure
	RBDTR	Right boundary temperature ratio
	RBDTRI	Initial right boundary temperature ratio
	RBNDRY	Denotes right boundary open or closed
	RNODE	Array for right most node to be corrected in CORRCT
R	RR	Q-A*S (extended Riemann variable)
	RXX	Node defining the right interval
S	S	Entropy
	SAP	Entropy to the left of the shock for flows right or entropy to the right of the C.S. for flows left
	SARRAY	Array of entropy for plotting
	SAVG	Average entropy
	SBP	Entropy to the right of the C.S. for flows right or entropy to the left of the shock for flows left
	SHKDIR	Shock direction of travel, 3 to the left, 2 to the right
	SHOCK	3 digit variable denoting shock location, direction of travel, and if it crosses a node
	SIGMA	Spatial location of discontinuities Sigma(L,J) where L indicates the type of discontinuity and J indicates the time level: 1--current level, 2--level being calculated
	SK	Integer that denotes relative location of shock near boundaries
	SKIP	Variable which indicates how many time steps between calls to output routines
	**STEP	The change in time of ** at a node used to step up to the next time level

t	T	Time since initial conditions
T	TEMP	Temperature
	TRI	Initial temp, ratio across the diaphragm
	TS	Time for shock to travel one interval
	T2	Temperature ratio at first node inside boundary
u_A	UA	Velocity relative to the shock, left side
u_B	UB	Velocity relative to the shock, right side
	VHEAD	Velocity of the head of the expansion wave for the exact solution
	VCDE	Velocity of the contact discontinuity for the exact solution
V_S	VS	The shock speed (positive right, negative left)
	VSE	Velocity of the shock for the exact solution
	VTAIL	Velocity of the tail of the expansion wave for the exact solution
w	W	Mach no. relative to a standard shock
$\bar{w}$		Vector of the principle variables, Q,R,S
X	X	Location in spacial plane (I-1)*H
	XARRAY	Array of spatial positions for plotting
	XEXACT	Array of six X values for the exact solution
	XINIT	Initial position of discontinuity for exact solution plotting
	X2	Location of node to right of discont. along the spacial axis
	Y	(N+1)/2
	YEXACT	Array of six density values for the exact solution

$\bar{Z}$	Z(K)	Vector of the right hand sides of the governing equations
Δ		Small spatial change
δ		Small change with respect to time
θ, ϕ		Flow angles with respect to reference coordinate planes

ACKNOWLEDGEMENT

Thanks to my best friend, Gayle LeVeque, for her support and encouragement, to Atul Mathur, for his invaluable assistance, and to Ray Shreeve, for making this a challenging and rewarding learning experience.

The work is dedicated in memory of my grandparents, Ray Litton Johnston and Anna Elizabeth Weech.

I. INTRODUCTION

The investigation of wave rotors[1] and their possible application in gas turbine engines at the Naval Postgraduate School's (NPS) Turbopropulsion Laboratory prompted the development of a one-dimensional unsteady flow computer code. The QAZ1D method developed by Verhoff [Ref. 1] was chosen over other methods [Ref. 2] for reasons which included the following:

1. The method is based on the use of characteristics. Such methods can model wave propagation accurately.

2. The use of a natural streamline coordinate system eases the difficult task of computing with two and three dimensional grids.

3. The equations are written in a form which allows a straightforward extension to viscous flows.

Salacka [Ref. 2] verified the development of the equations and implemented a one-dimensional Euler solution of the

---

[1]The wave rotor consists of simple tube-like passages along and around the periphery of a drum which rotates between end walls containing inlet and outlet (partial admission) gas ports. Compression and expansion processes are arranged to occur in a cyclically periodic unsteady process within the rotor such that a high pressure driver gas is used to compress a low pressure driven gas. The compressed driven gas is led from the outlet port to the combustor and brought back into the rotor as the driver. The expanded driver gas exits an outlet port. Thus the rotor functions as both turbine and compressor with direct gas-gas energy exchange through unsteady wave processes. The technology of such devices requires the design of appropriate cycles using one-dimensional calculations, and analysis of the multi-dimensional unsteady flows such as occur during port opening and closing.

shock tube problem in the EULER1 code. The EULER1 code was limited to high pressure set on the left, with flow and shock velocities to the right. The shock wave was tracked and corrected for in the flow, but the contact discontinuity was not, and the expansion wave was not tracked. The code also did not treat end boundary conditions.

The present work extended the EULER1 code of Salacka to become the "Euler 1D Version 2" or E1DV2 code which:

1) can handle both right and left traveling flow and discontinuities

2) implements tracking and correction of the contact discontinuity

3) implements tracking of the expansion wave

4) models closed wall and open end boundary conditions

5) models shock and contact surfaces within a grid interval.

Section II and Appendix A describe the analysis underlying the revisions. The Fortran code, E1DV2, is detailed in Section III and Appendix C and graphical results of four test cases to demonstrate the new features of the code are given in Section IV. A discussion of these results and limitations of the present code follow in Section V. Conclusions and recommendations are made in Section VI. Appendix B contains instructions to operate the code on the NPS computer and the FORTRAN listing is included as Appendix D.

II. ANALYTICAL AND COMPUTATIONAL APPROACH

A. QAZ1D DESCRIPTION

The QAZ1D analysis and numerical solution scheme is an explicit, non-conservative method that uses extended Riemann variables to model the multi-dimensional Euler equations in a natural streamline coordinate system. Table I lists a summary of the applicable equations which are derived in detail in [Ref. 2].

1. The Coordinate System

Figure 2.1 shows the natural streamline coordinate system (s,n,m) relative to a fixed rectangular cartesian system (x,y,z). The right-hand orthogonal system moves in curvilinear translation along a streamline. The "s" direction is measured in the direction of the flow, tangent to the streamline. The "n" direction is perpendicular to the s direction in a plane perpendicular to the x-z plane. The "m" direction is normal to the plane containing the n and s directions. The angle ϕ is the angle between the s-n and x-y planes, and the angle θ is the angle between the velocity vector and the x-z plane in the s-n plane. In the one-dimensional problem treated here, the s direction is such that velocity to the right is positive, and to the left is negative. [Ref. 1:p. 2]

TABLE I

SUMMARY OF THE EULER EQUATIONS

Definitions

Speed of sound $$A = \sqrt{\gamma P/\rho} \tag{I.1}$$

Modified entropy change $$dS = -\frac{dQ_R}{\gamma T} \tag{I.2}$$

so that

$$\frac{\bar{S}}{R_G} = S = \frac{1}{\gamma(\gamma-1)}[2\gamma - \ln(P/\rho^{\gamma})] \tag{I.3}$$

Extended Riemann variables

$$Q = q + AS \tag{I.4}$$

$$R = q - AS \tag{I.5}$$

Euler Equations

$$\frac{\partial Q}{\partial t} + (q+A)\frac{\partial Q}{\partial s} = -\frac{\gamma-1}{2}A(S - \frac{2}{\gamma-1})[\frac{\partial}{\partial s}(q - \frac{2}{\gamma-1}A)] - \frac{\gamma-1}{2}qAS[\frac{\partial\theta}{\partial n} + \cos\theta\frac{\partial\phi}{\partial m}] \tag{I.6}$$

$$\frac{\partial R}{\partial t} + (q-A)\frac{\partial R}{\partial s} = \frac{\gamma-1}{2}A(S - \frac{2}{\gamma-1})[\frac{\partial}{\partial s}(q + \frac{2}{\gamma-1}A)] + \frac{\gamma-1}{2}qAS[\frac{\partial\theta}{\partial n} + \cos\theta\frac{\partial\phi}{\partial m}] \tag{I.7}$$

$$\frac{\partial S}{\partial t} + q\frac{\partial S}{\partial s} = \phi \tag{I.8}$$

$$\frac{\partial\theta}{\partial t} + q\frac{\partial\theta}{\partial s} = -\frac{A^2}{\gamma q}\frac{\partial \ln P}{\partial n} \tag{I.9}$$

$$\frac{\partial\phi}{\partial t} + q\frac{\partial\phi}{\partial s} = -\frac{A^2}{\gamma q\cos\theta}\frac{\partial \ln P}{\partial m} \tag{I.10}$$

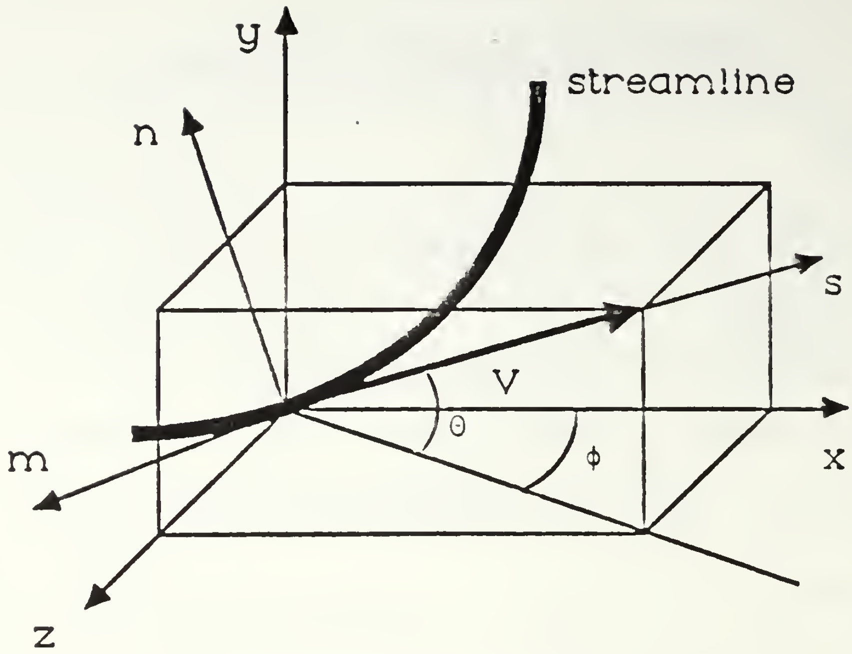


Figure 2.1 The Natural Coordinate System

2. Extended Riemann Variables

The "Riemann Variables" in most publications are defined as

$$R_1 = q - \left(\frac{2}{\gamma+1}\right) A$$

and (2.1)

$$R_2 = q + \left(\frac{2}{\gamma+1}\right) A$$

where q is the velocity magnitude, A is the speed of sound, and γ is the ratio of the specific heat at constant pressure to the specific heat at constant volume for a particular gas

[Ref. 3:p. 5]. Verhoff modified the definition to obtain "Extended Riemann Variables," defined as

$$Q = q + AS \tag{I.4}$$

$$R = q - AS \tag{I.5}$$

where S is the modified entropy given in Table I [Ref. 1:p. 2].

3. <u>The Governing Equations</u>

The Euler equations in Table I were developed in detail [Ref. 2:Appendix A] from the equations for conservation of mass, momentum, and energy for a control volume. The underlying assumptions comprise inviscid flow, negligible body forces, no heat transfer, and a perfect gas. These assumptions and the definition of modified entropy were used in transforming the equations into a form suitable for solution using the method of characteristics.

B. CHARACTERISTIC SOLUTION

Equations (I.6), (I.7), (I.8), (I.9) and (I.10) are a system of slightly coupled quasi-one-dimensional partial differential equations (PDE) with eigenvalues $q+A$, q, and $q-A$. These equations can be rewritten along the characteristics in the time-space domain to become ordinary differential equations (ODE) which are solved by quadrature and interpolation. The propagation of each characteristic

curve along its particular trajectory is expressed by the respective ODE. [Ref. 1:p. 1]

Each in the system of equations is in the form

$$\frac{\partial w}{\partial t} + \lambda \frac{\partial w}{\partial s} = z \tag{2.2}$$

If w = w(s,t) it is shown by Salacka [Ref. 2:pp. 20-23] that

$$\lambda = \frac{ds}{dt} \tag{2.3}$$

and

$$\frac{dw}{dt} = z \tag{2.4}$$

where λ describes the characteristic curve along which w changes.

In this section the solution of the 3D Euler equations given in Table I is described. The computer code developed in the present work, however, was for the 1D case wherein equations (I.9) and (I.10) are not required, and only the three remaining equations (I.6), (I.7), and (I.8) are considered.

Equations (I.6) to (I.8) for one-dimensional flow may be expressed in matrix form by setting

$$\bar{w} = \begin{bmatrix} Q \\ R \\ S \end{bmatrix} \qquad [\lambda] = \begin{bmatrix} q+A & 0 & 0 \\ 0 & q-A & 0 \\ 0 & 0 & q \end{bmatrix}$$

and

$$\bar{z} = \begin{bmatrix} -\frac{\gamma-1}{2}A(S-\frac{2}{\gamma-1})\,[q-\frac{2}{\gamma-1}A]_s \\ \frac{\gamma-1}{2}A(S-\frac{2}{\gamma-1})\,[q+\frac{2}{\gamma-1}A]_s \\ 0 \end{bmatrix}$$

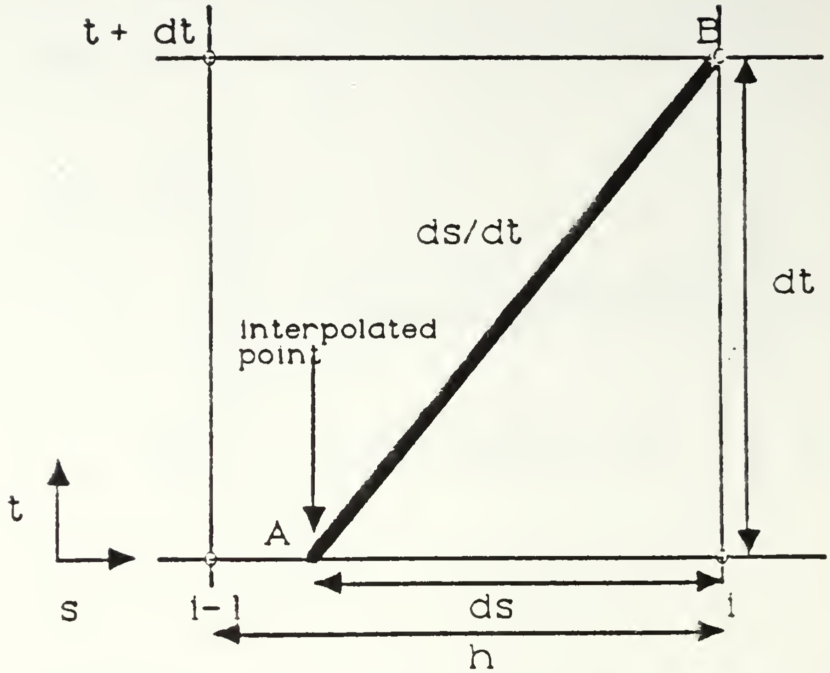
Then equation (2.2) becomes, for the equation set,

$$\frac{\partial\bar{w}}{\partial t} + [\lambda]\,\frac{\partial\bar{w}}{\partial s} = \bar{z}\,.$$

Figure 2.2 shows the characteristic curve in the space-time domain between two nodes within the computational mesh. Equation (2.2) has a solution

$$\delta\bar{w} = -\Delta\bar{w} + \int_t^{t+dt} \bar{z}\,dt \tag{2.5}$$

where $\delta\bar{w}$ is the change due to time at a fixed location, and $\Delta\bar{w}$ is the change due to displacement at a fixed location, along the characteristic curve. The value of $\Delta\bar{w}$ is calculated by interpolation through an iterative scheme



$$\delta\bar{w} = \bar{w}_B - \bar{w}_{s_i}$$

$$\Delta\bar{w} = \bar{w}_{s_i} - \bar{w}_A$$

Figure 2.2 Characteristic Curve Within Computational Mesh

using initial guesses for the characteristic slope, λ, from values at time level t. The line integral is approximated by

$$\int_t^{t+dt} \bar{z}\, dt = \int_A^B \frac{\bar{z}}{\lambda}\, ds = \int_{S_i-\Delta S}^{S_i} \frac{\bar{z}}{\lambda}\, ds = \frac{\bar{z}}{\lambda}(S_i-(S_i-\Delta S))$$

$$= \bar{z}(\Delta s/\lambda)$$

$$= \bar{z}(\delta t) \tag{2.6}$$

The spatial derivatives of q and A in $\overline{z}$ are estimated from values at A and s_i. Finally, $\overline{\partial w}$ is calculated, and each node is updated to the next time level.

The Courant-Friedrichs-Lewy (CFL) convergence condition requires the domain of dependence of the numerical approximation to include the domain of dependence of the differential equation [Ref. 4:pp. 336-337]. Figure 2.3 shows that

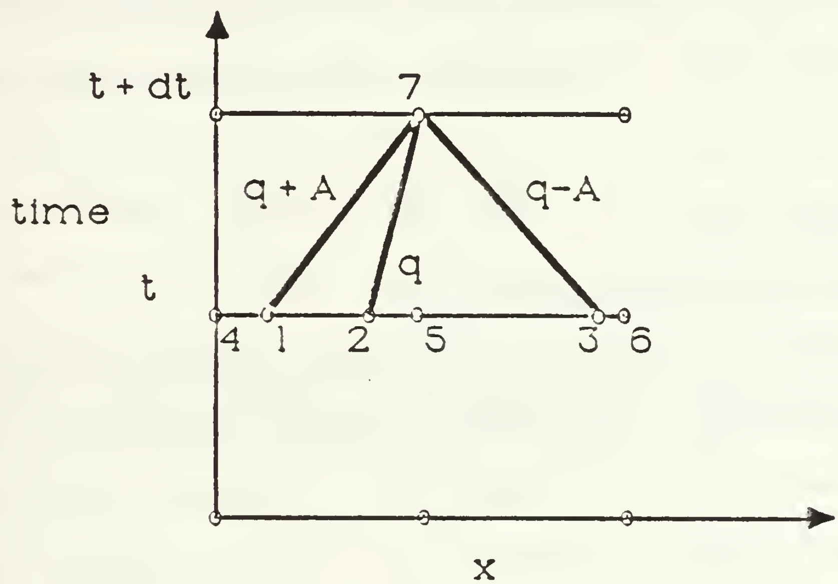


Figure 2.3 Computational Grid for the QAZ1D Method

the computational grid for the QAZ1D method satisfies this condition by having interpolated initial data points (points 1, 2, and 3) in between previous solution nodes (points 4, 5, 6). The q+A characteristic is estimated from values at point 4, q characteristic from values at point 5, and q-A characteristic from values at point 6. The use of a maximum

time step keeps the domain of dependence of the numerical scheme just outside that of the actual equations, aiding convergence. Unlike finite difference schemes that are conservative and require a numerical diffusion or viscosity to handle oscillations or smearing, this method is stable without numerical smoothing or artificial dissipation [Ref. 5:pp. 562-583]. However, the equations do not account for the discontinuities across shocks and contact surfaces properly. Thus after each time step a one-point correction technique is used to correct for shocks and contact surfaces.

C. DISCONTINUITIES

1. Expansion/Rarefaction Waves

The head and tail of the expansion wave travel along the characteristic as gradient discontinuities. Flow velocity, speed of sound, pressure, and density are continuous across a gradient discontinuity but the spatial derivatives are discontinuous. Entropy remains constant through the rarefaction wave. The head of an expansion wave moves into undisturbed flow with a speed $q+A$, while the tail of the expansion wave has a velocity $q-A$. No special treatment of these waves is required; the characteristics method accurately tracks and models their influence. Expansion waves are generated when two shocks collide, a diaphragm is burst, a contact surface and shock intersect, a shock leaves

an open boundary, or an open boundary has a pressure lower than the pressure inside the tube. [Ref. 3:pp. 13-15]

2. Shocks

Shocks are created when a diaphragm is burst, compression waves generated at an open boundary coalesce into a wavefront with sufficient strength, or when a shock and contact surface collide. Pressure, velocity, density, and entropy are discontinuous across a shock [Ref. 6:pp. 30-31]. The equations of motion (I.6)-(I.10) are not correct for calculating changes through shocks. A method to correct for a shock is to use the ratio of the jump in the extended Riemann variable across the shock to the speed of sound downstream of the shock to find the incoming Mach Number relative to the shock (w) [Ref. 1:p. 3]. Figure 2.4 shows the case of a shock headed to the right. The shock velocity

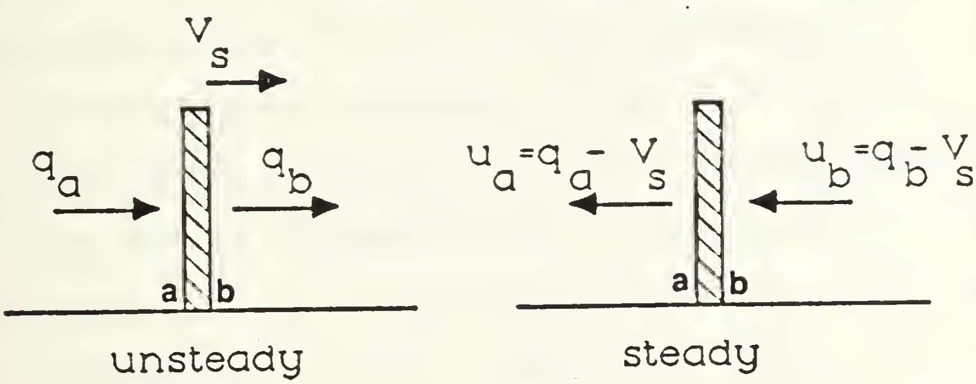


Figure 2.4 Shock Wave with High Pressure on Left

(V_s) is imposed on the flow to produce a stationary shock condition. Flow variables are then corrected using normal shock relations. Figure 2.4 applies to a condition where high pressure is on the left. The shock is moving to the right at V_s into flow with velocity q_B. Table II lists the equations developed by Salacka [Ref. 2] for relating the extended Riemann variable change to Mach Number. Velocities are non-dimensionalized by the speed of sound downstream of the shock direction, A_B, and pressures and densities by the respective values downstream.

Figure 2.5 depicts equation (II.2) over a range of Mach Numbers from 1.0 to 4.0. The curve is approximated by the quadratic equation

$$\frac{Q_A - Q_B}{A_B} = -2.7574 + 3.1573w - 0.2863w^2 \qquad (2.7)$$

If the high pressure is to the right the shock moves as Fig. 2.6 illustrates, and the governing equations are as listed in Table III. Again, downstream conditions are used to non-dimensionalize velocity, pressure, density, and the Riemann variable change. The development of the equations in Table III is given in Appendix A. A plot of equation (III.2) over a range of w from 1.0 to 4.0 is shown in Fig. 2.7. The curve is approximated by the quadratic equation

$$\frac{R_B - R_A}{A_A} = 2.7574 - 3.15732w + 0.2863w^2 \qquad (2.8)$$

TABLE II

SHOCK EQUATIONS FOR HIGH PRESSURE ON LEFT

Definitions

Relative Incoming Mach Number:

$$w = -\frac{u_B}{A_B} = -\frac{(q_B - V_s)}{A_B} \tag{II.1}$$

Equations

Q Riemann Variable Change:

$$\frac{Q_A - Q_B}{A_B} = \frac{2(w^2-1)}{(\gamma+1)w} + \left(\frac{2}{\gamma-1}\right)\left[\frac{A_A}{A_B} - 1\right] - \frac{A_A}{A_B}\left[\frac{1}{\gamma(\gamma-1)}\right]\ln\left[\left(\frac{2\gamma}{\gamma+1}w^2 - \left(\frac{\gamma-1}{\gamma+1}\right)\right)\left(\frac{(\gamma-1)w^2+2}{(\gamma+1)w^2}\right)^{\gamma}\right] \tag{II.2}$$

Speed of Sound Ratio:

$$\frac{A_A}{A_B} = \frac{1}{(\gamma+1)w}\left[2(\gamma-1)\left[1 + \frac{\gamma-1}{2}w^2\right]\left[\frac{2\gamma}{\gamma-1}w^2 - 1\right]\right]^{1/2} \tag{II.3}$$

Pressure Ratio:

$$\frac{P_A}{P_B} = \frac{2\gamma}{\gamma+1}w^2 - \frac{\gamma-1}{\gamma+1} \tag{II.4}$$

Density Ratio:

$$\frac{\rho_A}{\rho_B} = \frac{(\gamma+1)w^2}{(\gamma-1)w^2+2} \tag{II.5}$$

TABLE II (CONTINUED)

Velocity Change:

$$\frac{q_A - q_B}{A_B} = \frac{2(w^2-1)}{(\gamma+1)w} \tag{II.6}$$

Entropy Change:

$$S_A - S_B = -\left(\frac{1}{\gamma(\gamma-1)}\right)\ln\left[\frac{2\gamma w^2-\gamma+1}{\gamma+1}\right] - \left(\frac{1}{\gamma-1}\right)\ln\left[\frac{(\gamma-1)w^2+2}{(\gamma+1)w^2}\right] \tag{II.7}$$

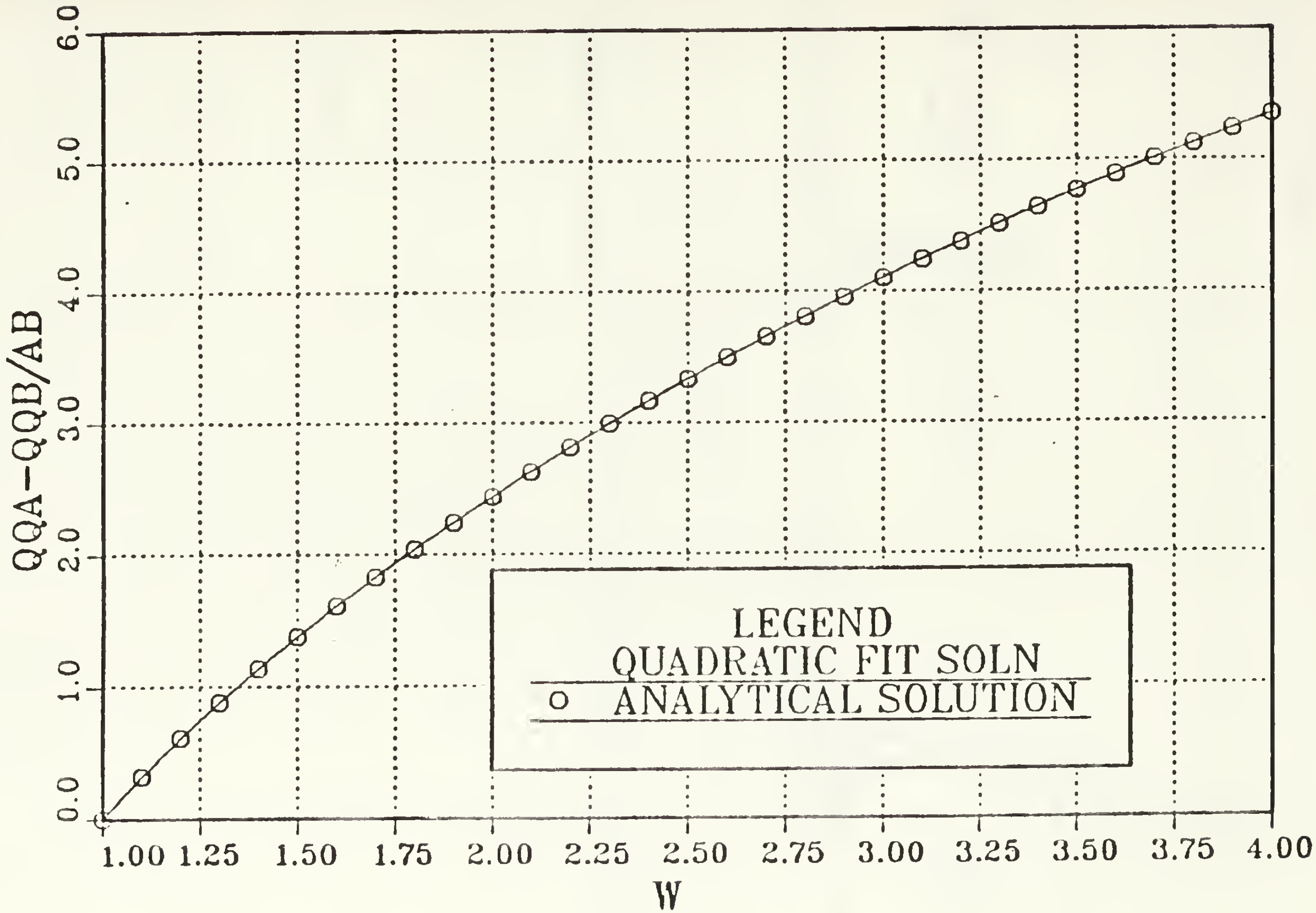


Figure 2.5 Q, Extended Riemann Variable Change with Mach Number

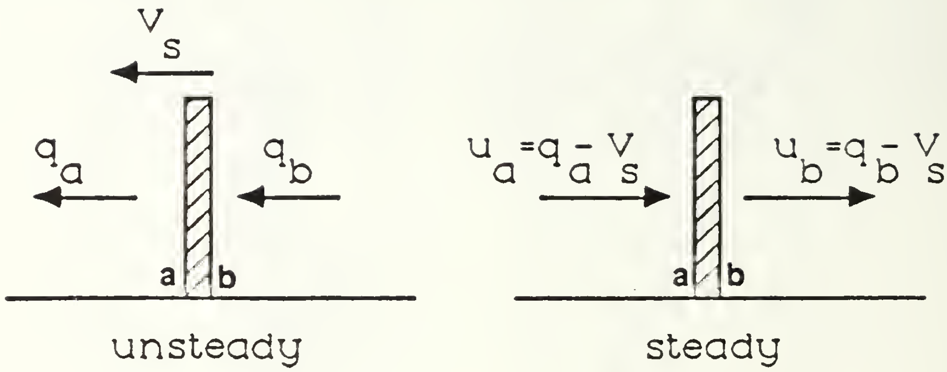


Figure 2.6 Shock Wave with High Pressure on Right

TABLE III

SHOCK EQUATIONS FOR HIGH PRESSURE ON RIGHT

Definitions

Relative Incoming
Mach Number:

$$w = \frac{u_A}{A_A} = \frac{q_A - V_s}{A_A} \qquad \text{(III.1)}$$

Equations

R Riemann
Variable Change:

$$\frac{R_B - R_A}{A_A} = \frac{2(1-w^2)}{(\gamma+1)w} + \left(\frac{2}{\gamma-1}\right)\left[1 - \frac{A_B}{A_A}\right] + \frac{A_B}{A_A}\left[\frac{1}{\gamma(\gamma-1)}\right]\ln\left[\left(\frac{2\gamma}{\gamma+1}w^2 - \left(\frac{\gamma-1}{\gamma+1}\right)\right)\left(\frac{(\gamma-1)w^2+2}{(\gamma+1)w^2}\right)^{\gamma}\right] \qquad \text{(III.2)}$$

Speed of Sound Ratio:

$$\frac{A_B}{A_A} = \frac{1}{(\gamma+1)w}\left[2(\gamma-1)\left[1 + \frac{\gamma-1}{2}w^2\right]\left[\frac{2\gamma}{\gamma-1}w^2 - 1\right]\right]^{1/2} \qquad \text{(III.3)}$$

Pressure Ratio:

$$\frac{P_B}{P_A} = \frac{2\gamma}{\gamma+1}w^2 - \frac{\gamma-1}{\gamma+1} \qquad \text{(III.4)}$$

Density Ratio:

$$\frac{\rho_A}{\rho_B} = \frac{(\gamma+1)w^2}{(\gamma-1)w^2+2} \qquad \text{(III.5)}$$

Velocity Change:

$$\frac{q_B - q_A}{A_A} = \frac{2(1-w^2)}{(\gamma+1)w} \tag{III.6}$$

Entropy Change:

$$S_B - S_A = -\left(\frac{1}{\gamma(\gamma-1)}\right)\ln\left[\frac{2\gamma w^2 - \gamma + 1}{\gamma+1}\right] - \left(\frac{1}{\gamma-1}\right)\ln\left[\frac{(\gamma-1)w^2+2}{(\gamma+1)w^2}\right] \tag{III.7}$$

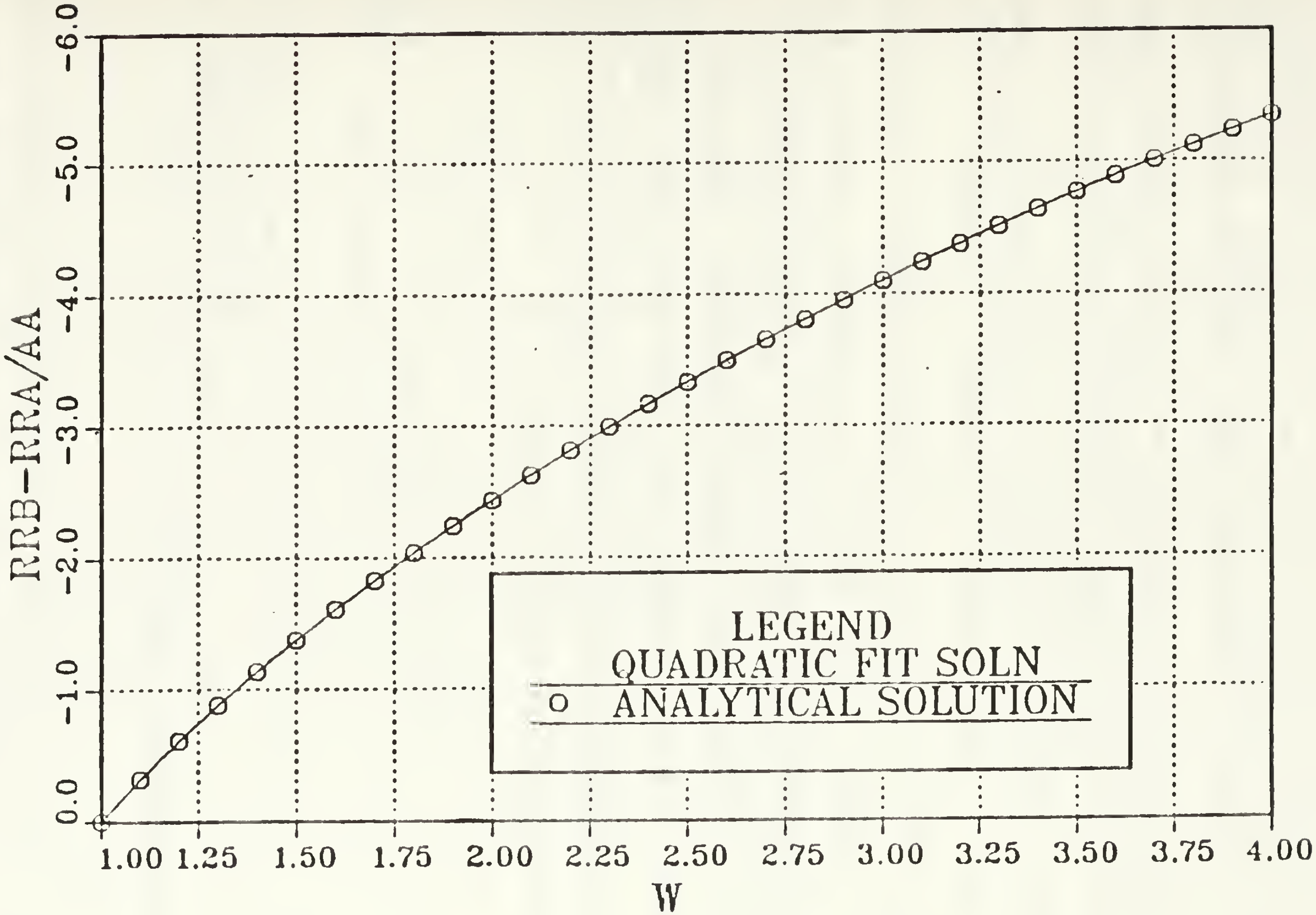


Figure 2.7 Extended Riemann Variable Change with Mach Number, High Pressure on Left

An examination of equations (II.2) and (III.2) reveals that

$$\frac{R_B - R_A}{A_A} = -\left[\frac{Q_A - Q_B}{A_B}\right]$$

This is expected since in flow to the right, the Q extended Riemann variable is associated with the $q+A$ characteristic. Also, since velocity is defined as positive to the right, q is positive. But for flow to the left, velocity is defined as negative, thus q is negative. Moretti [Ref. 3] showed that the downstream running Riemann variable ($q+A$ characteristic) has a lower magnitude of discontinuity across a normal shock than the upstream running Riemann variable ($q-A$ characteris-tic), and is thus preferable. In flow to the left, the Riemann variable associated with downstream is the R Riemann variable, though the characteristic associated with the R Riemann variable is $q-A$, in flow to the left q is negative and thus

$$q - A = -(|q| + A) \tag{2.9}$$

Therefore, in fluid traveling left the R Riemann variable is used to find Mach Number, and in fluid traveling right the Q Riemann variable is used.

If the shock is between two nodes with no contact surface within the same interval, the method to calculate the Mach Number, w, is as follows:

1) The change in the appropriate extended Riemann variable across the interval is measured; R variable for flow left, and Q variable for flow right. Call this change, Δm.

2) Use Δm in equation (2.7) or equation (2.8) for the appropriate flow to calculate w.

3) With equation (II.2) or equation (II.3), for flow right or left respectively, use w to calculate the analytical Riemann variable change, Δe.

4) If this exact value of Δe is not equal to Δm then calculate a new guess, Δg from

$$\Delta g_{i+1} = \Delta g_i + (\Delta m - \Delta e) \tag{2.10}$$

and then repeat steps 2 to 4 until Δe equals Δm.

Thus the correct value of w is determined, and shock corrections from the normal shock relations are valid. V_s is determined from equation (II.1) or equation (III.1) depending on flow direction. Flow direction will be initially determined by the side with high pressure. High pressure on the left means flow will travel to the right. Likewise, high pressure on the right means flow will travel to the left.

3. <u>Contact Surface</u>

A contact surface is a boundary between regions of flow which are different in composition or which have undergone different thermodynamic histories. Thus a contact surface is formed when the diaphragm is burst in a shock

tube separating the gas initially in the driver from the gas initially in the driven tube [Ref. 6:pp. 23-29]. In a wave rotor, a contact surface would separate the combustor gases from the cooler inlet air. Contact surfaces are also caused by two shocks of opposite families colliding or a shock over taking a shock of the same family [Ref. 3:pp. 15-18].

Figure 2.8 illustrates the changes in physical properties through the contact surface in the shock tube problem. The gas to the right has been compressed and heated by the shock wave, with that to the left cooled and

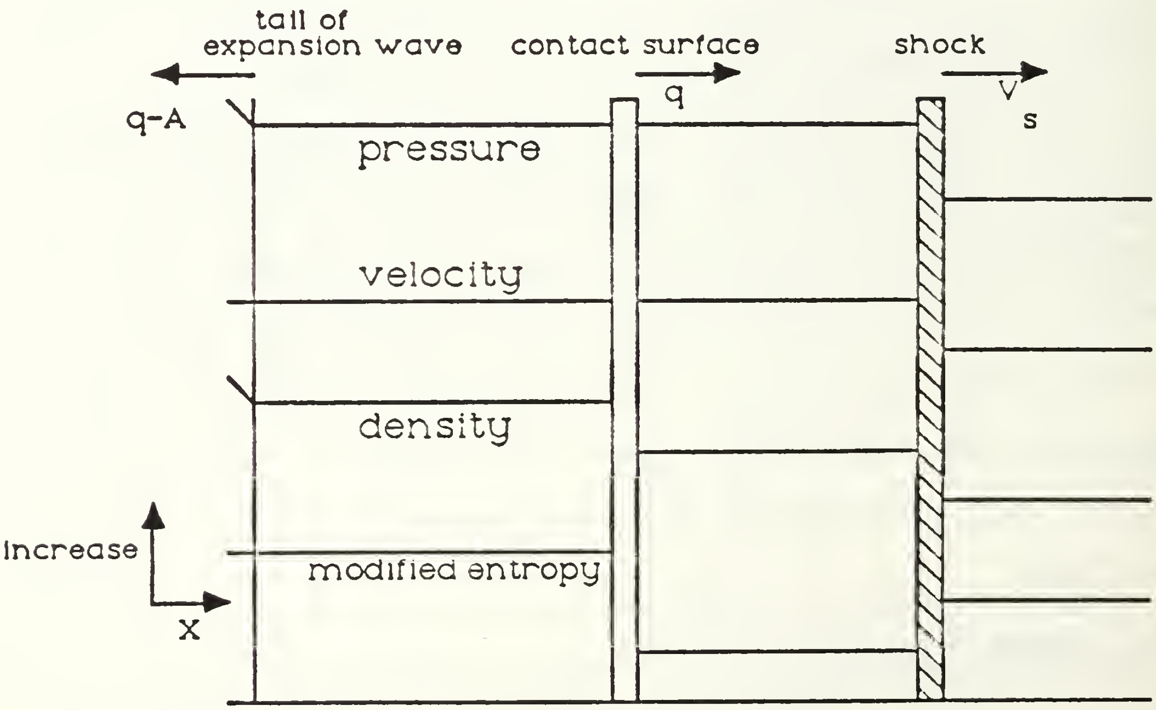


Figure 2.8 Behavior of Physical Properties Through a Contact Surface

expanded. The density, speed of sound, and modified entropy are discontinuous across the contact surface.

Since pressure is constant across the contact surface, using the perfect gas equation of state and the first law of thermodynamics it is shown in Appendix A for the shock tube problem that

$$A_A/A_B = \exp^{\left(\frac{(\gamma-1)}{2}(S_B-S_A)\right)} \tag{2.11}$$

Since the flow behind the contact surface to the tail of the rarefaction wave is isentropic, the value of S_B will be known. The value of S_A is the value of entropy behind the shock. Subtracting equations (I.4) and (I.5) from each other and dividing by entropy S_B, A_B can be determined. Thus the unknown speed of sound A_A is obtained from equation (2.11). With q_B equal to q_A, and S_A the Riemann variables just behind the contact surface are calculated from equations (I.4) and (I.5). For a contact surface traveling to the left it is only necessary to interchange the subscripts. The velocity of the contact surface is easily determined since it moves at velocity q along the q characteristic line associated with S. [Ref. 3:p. 16]

4. <u>Contact Surface/Shock Interaction</u>

When the shock and contact surface are within the same interval the calculation of the shock incoming Mach Number, w, must be modified. Figure 2.9 shows a plot of the

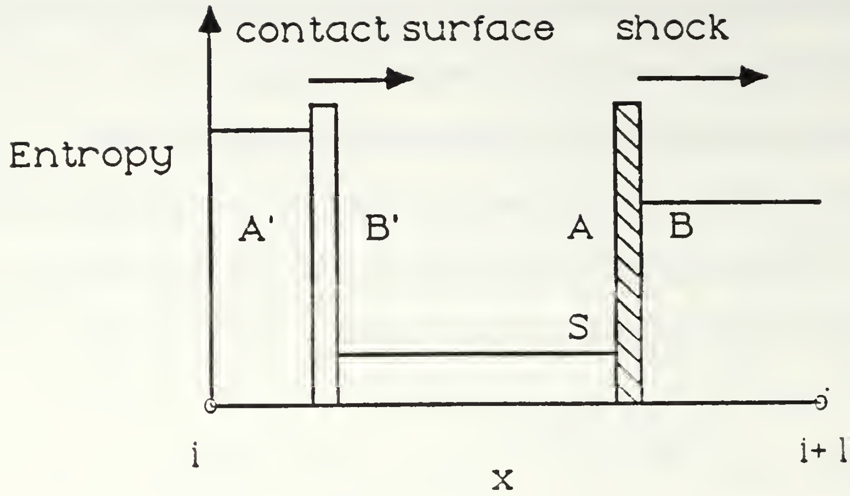


Figure 2.9 Modified Entropy Distribution for Shock and Contact Surface Traveling Right

modified entropy change across the interval. The use of the Riemann values at the nodes to estimate the Riemann variable change across the shock would be incorrect. The correct Riemann variable change would be, for flow to the right,

$$\Delta SK = \frac{Q_S - Q_B}{A_B} \qquad (2.12)$$

The correct Mach Number, w, is determined using the technique of Moretti modified for QAZ1D [Ref. 3:pp. 23-24].

Referring to Fig. 2.9, the technique for flow to the right is as follows:

1) Calculate w as if no contact surface were in the interval. With values at node i+1 known, determine S_B. Then use w and S_B in equation (II.7) to solve for S_A. This is the initial estimate of S_S. Let $S_{B'} = S_S$.

2) The Q Riemann Variable change across a contact surface is shown in Appendix A to be given by

$$\Delta CS = \frac{Q_{A'}-Q_{B'}}{A_B} = [e^{[(S_{B'}-S_{A'})(\frac{\gamma-1}{2})]} S_{A'}-S_{B'}] \frac{A_A}{A_B} \tag{2.13}$$

3) Define

$$\Delta m = \frac{Q_{A'}-Q_B}{A_B} \tag{2.14}$$

then

$$\Delta m = \Delta SK + \Delta CS \tag{2.15}$$

can be solved to obtain ΔSK.

4) Using ΔSK as Δm in the shock iteration scheme given in Section II.C.2 calculate a new w.

5) With this new w and S_B use equation (II.7) to obtain a new S_A.

6) Compare S_A with S_S. If they are not equal to within an acceptable error, calculate a new estimate for S_S using

$$S_{S_{i+1}} = S_{S_i} + (S_A - S_S) \tag{2.16}$$

Iterate until convergence is achieved.

This will result in the proper w for a condition with shock and contact surface interacting.

For flow to the left, it is only necessary to interchange the subscripts A and B, and use

$$\frac{R_{B'}-R_{A'}}{A_A} = [\exp^{(\frac{(\gamma-1)}{2}(S_{A'}-S_{B'})} S_{B'}-S_{A'}] (\frac{A_B}{A_A}) \tag{2.17}$$

for the extended Riemann Variable change. Figure 2.10 illustrates the left traveling condition.

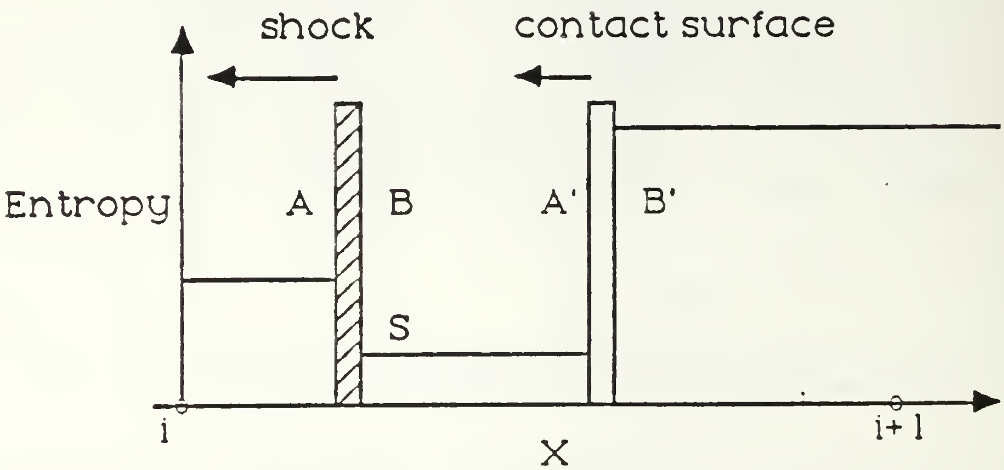


Figure 2.10 Modified Entropy Distribution for Shock and Contact Surface Traveling Left

D. BOUNDARY CONDITIONS

1. Open Boundary

At the open boundary, a reference pressure ratio, $P_{ref} = P_\infty/P_A$ is specified, where P_∞ is the pressure to be held at the boundary, and P_A is the pressure just inside the tube. Only the case where $P_\infty \leq P_A$ is considered. This means that an outflow condition at the boundary will result when $P_\infty < P_A$, with an expansion wave traveling in from the boundary. Thus locally at the boundary, conditions are isentropic.

The computational grid for the left boundary is shown in Fig. 2.11. A phantom node, L, is used outside the

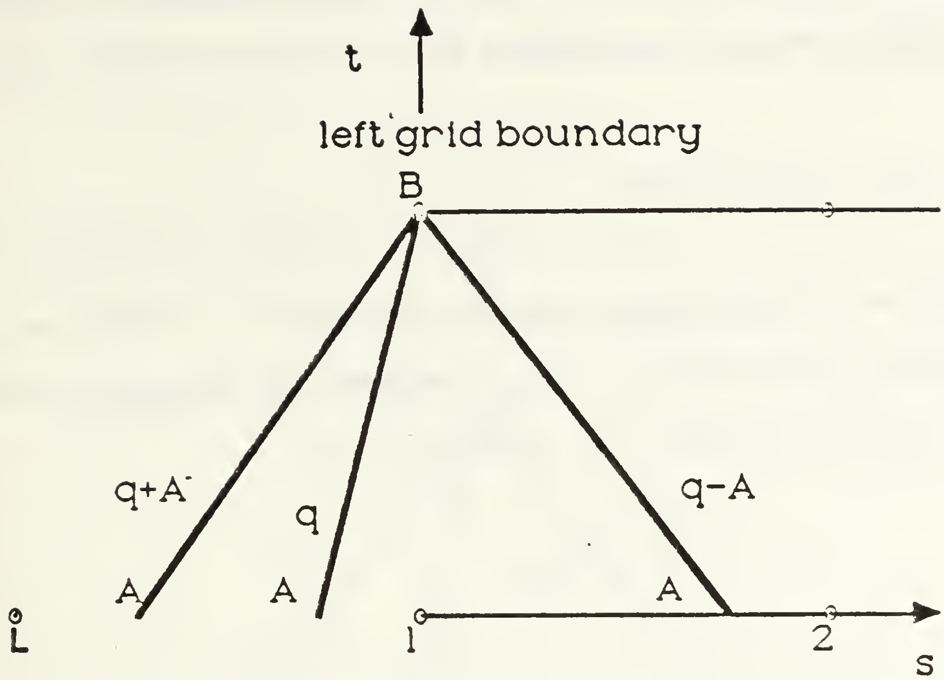


Figure 2.11 Left Boundary Computational Grid

computational mesh to enable simple enforcement of constant pressure and entropy at the mesh boundary node, 1.

The approach is to determine the required variables at node L, then transfer these values onto node 1, and vary q at the boundary to meet the required boundary conditions. First, using P_{ref} at node L and S at node 2 in equation (I.3) rewritten in the form

$$\rho_R = ((1/P_{ref})(\exp^{(S - \frac{2}{\gamma-1})(-\gamma(\gamma-1))}))^{-1/\gamma} \tag{2.18}$$

ρ_R, the density ratio at the boundary, is calculated. Using the perfect gas equation of state in equation (2.18) the temperature ratio, T_R is given by

$$T_R = (\rho_R^{(\gamma-1)})(\exp^{(\gamma(1-\gamma)(S-\frac{2}{\gamma-1})}) \tag{2.19}$$

and can be calculated once ρ_R is known. With an initial velocity at the boundary, q_b, assumed the Riemann variables R and Q are determined for node L using

$$R = q_b - \sqrt{T_R}\, S \tag{2.20}$$

and

$$Q = q_b + \sqrt{T_R}\, S \tag{2.21}$$

Subtracting equation (2.20) from (2.21), and dividing by twice S yields A at node L, where $A = \sqrt{\gamma R_G T}$. The values of R, Q, A, q_b, and S are now known at node L. These values are assigned to node 1, and the QAZ1D algorithm is applied to the boundary node as if it were an interior node. The result is an estimate of the new values for R, Q, and S at node 1. These are used to obtain a new estimate of the pressure ratio at node 1, call it P_{RN}. P_{RN} is compared with P_{ref}, and the new value of S with the old value of S at node 1. A new q_b is calculated by solving equations (2.20)

and (2.21) simultaneously. The process is repeated until entropy and pressure remain constant at the boundary. For the right-hand boundary, Fig. 2.12 shows the computational

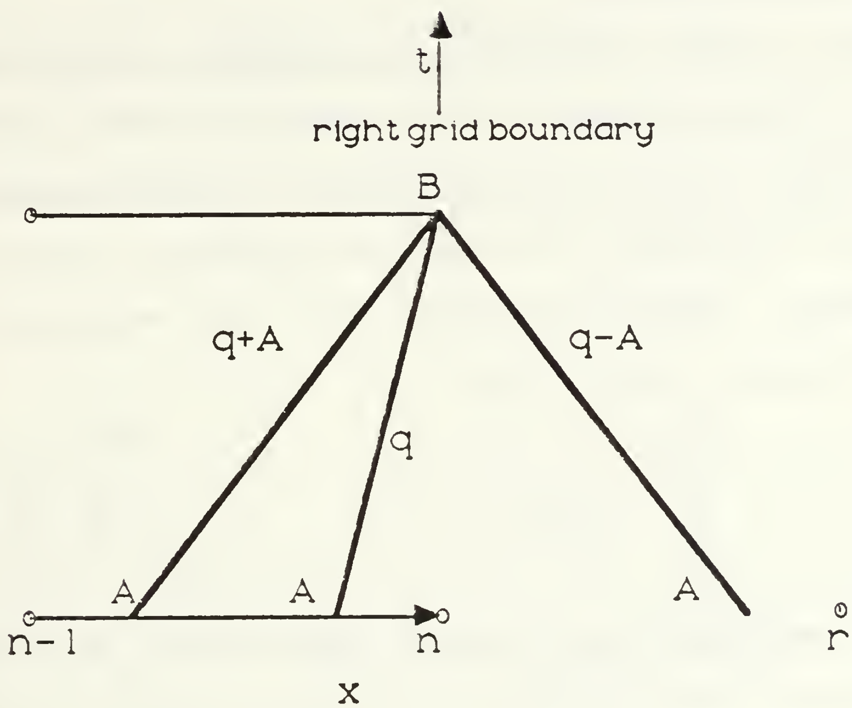


Figure 2.12 Right Boundary Computational Grid

grid with the phantom node, r, to the right of the mesh boundary node, n. The procedure for the right boundary is the same as for the left boundary.

When a shock travels across the open boundary, the boundary is first corrected using the method described in Section II.C.2. Subsequently, the pressure is returned to give the constant value specified for P_{ref}, using the above

procedure. The result will be an expansion wave traveling inward.

For supersonic flow, all of the variables are determined from values at the interior nodes [Ref. 1:p. 3].

2. Closed Boundary

The solid boundary is imposed by setting the velocity at the boundary, q, equal to zero. The same computational grid is used as that for an open boundary, but a different technique is used in assigning values to the phantom node. Referring to Fig. 2.11, the velocity at node 2, q_2, is known. By setting

$$q_L = - q_2 \tag{2.22}$$

the wave impacting the boundary will be met by a wave of equal velocity but opposite in direction and will result in node 1 appearing as a solid boundary. Further, to facilitate the QAZ1D method, set

$$R_L = - Q_2 \tag{2.23}$$

$$Q_L = |R_2| \tag{2.24}$$

$$S_L = S_2 \tag{2.25}$$

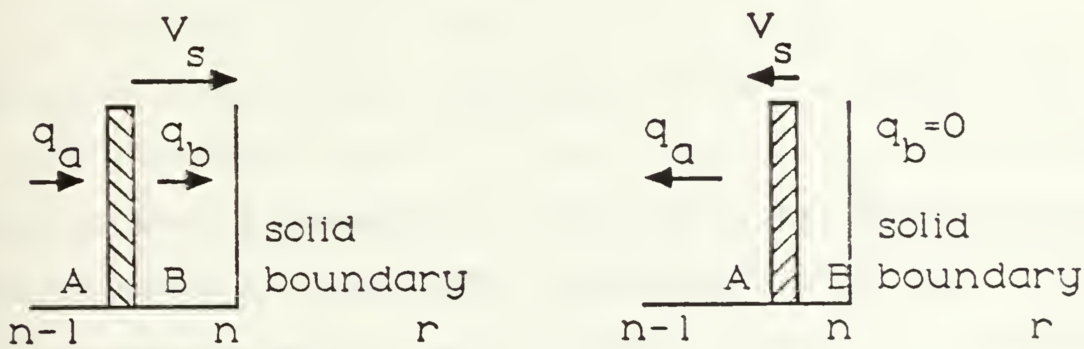
$$A_L = 2 \cdot A \tag{2.26}$$

This will enable the computation of the new boundary node 1 values as an interior point with

$$q_1 = 0$$

The right boundary is handled in the same way.

Figure 2.13 illustrates the sequence of events when a shock reflects off a solid boundary on the right [Ref. 7:



A) Shock Approaches Solid Boundary

B) Shock Reflects Off Solid Boundary

Figure 2.13 Shock Reflecting at Solid Boundary

pp. 172-195]. Since the velocity behind the reflected shock, q_B, is zero, and the velocity in front of the shock, q_A, is known, the velocity gradient, Δq, over the shock is, in non-dimensionalized form, given by

$$\Delta q = (q_B - q_A)/A_A \tag{2.27}$$

Rearranging equation (III.6), and using $\gamma = 1.4$ it can be shown that

$$w = \sqrt{1 + 0.36(\Delta q)^2} - 0.6\Delta q \tag{2.28}$$

This is an exact analytical value for w based on stationary normal shock relations. The speed of sound ratio, A_B/A_A, is calculated from equation (III.3) and

$$A_B = A_A(A_B/A_A) \tag{2.29}$$

Entropy behind the shock, S_B, is determined from equation (III.7) since S_A is known. Then the extended Riemann variables, R and Q, behind the reflected shock are calculated from equations (I.4) and (I.5) respectively.

For a shock reflected from the left boundary, the subscripts A and B are interchanged and using equation (II.6) the Mach Number is given by

$$w = \sqrt{1 + 0.36(\Delta q)^2} + 0.6\Delta q \tag{2.30}$$

The equations from Table II are used instead of those from Table III (used in the case of reflection from the right boundary), since the shock is now traveling from left to right.

III. FORTRAN PROGRAM E1DV2

A. GENERAL DESCRIPTION

Program E1DV2 is a Fortran computer program which calculates 1-dimensional unsteady flow based on the QAZ1D formulation and method of solution of the Euler equations of motion. The program has provisions for tracking in time the location of shock waves, contact surfaces and head and tail of the expansion waves. Its intended application to wave-rotor design and analysis explains the present definition of node points and boundary conditions. In its current form, the code describes flow in a tube initiated by a diaphragm bursting. The location of the diaphragm, whether the flow is to the left or the right, and whether the ends of the tube are closed or open, can be varied.

The E1DV2 program is written in FORTRAN 77, and was developed using the IBM 3033 System 370 computer. The use of DISSPLA subroutines in plotting results requires compiling and executing using VS FORTRAN procedures. Appendix B describes the procedures for running the E1DV2 program on the Naval Postgraduate School IBM computer system. A listing of the code, which contains its own description in comment statements, is given in Appendix D.

The code incorporates:

1) first order accuracy
2) double precision numerics, except for single precision in graphical output
3) linear interpolation and extrapolation algorithms
4) quadratic polynomial approximations for curve fitting
5) non-dimensional variable input and output
6) structured subroutine format.

Table IV lists all the subroutines called from the main program and brief descriptions of their purposes. The extensive use of subroutines was intended to allow modifications and extensions of the code without major restructuring.

B. THE MAIN PROGRAM

The main program flowchart is illustrated in Appendix C, Fig. C.1. The conventions and a comprehensive list of variables are listed in the beginning of the program. The number of grid nodes is set by the user, and must be an odd number. Arrays are dimensioned to the number of nodes in the main program prior to compiling and executing.

Initial values for temperature, pressure, and density ratios at the diaphragm and boundaries are entered by editing. Also, the initial velocity distribution on each side of the diaphragm is set, and as well as the ratio of specific heats. The side with the high pressure is identified. Boundaries are set either closed or open. In the

TABLE IV

SUBROUTINES

TIME	Calculates minimum time step
TRAK	Tracks discontinuities, calculates shock mach number, w
SWEEP	Advances node values to next time step
COND1	Calculates interpolated values of Riemann variables, $\partial q/\partial s$, and $\partial A/\partial s$ when no shock or contact surface within an interval
COND2	Calculates interpolated values of Riemann variable, $\partial q/\partial s$, and $\partial A/\partial s$ when discontinuity within interval to right of node or flow is supersonic to the right
COND3	Calculates interpolated values of Riemann variable, $\partial q/\partial s$, and $\partial A/\partial s$ when discontinuity within interval to left of node or flow is supersonic to the left
COND4	Called when node is jumped by discontinuity to load node information into array for use in subroutine CORRCT
COND5	Calculates interpolated values of Riemann variable, $\partial q/\partial s$, and $\partial A/\partial s$ when discontinuity on either side of node and flow is supersonic
COND6	Would contain subroutine to advance node when contact surface is in front of shock after intersecting and traveling in same direction. Not currently in use
COND7	Would contain subroutine to solve precisely when and where shock and contact surface would intersect within an interval. Not currently in use
COND7N	Would contain subroutine to solve precisely when and where shock and contact surface would intersect if either were jumping a node simultaneously. Not currently in use

TABLE IV (CONTINUED)

COND7S	Would contain the subroutine to solve the Riemann problem of shock and contact surface intersecting. Not currently in use
COND8	Would contain subroutine to advance node after shock and contact surface have crossed and are moving apart. Not currently in use
CORRCT	Advances nodes jumped by discontinuity to next time step
DELTAX	Determines relative location of discontinuity within an interval
SKJUMP	Calculates variable change across a normal stationary shock
CSJUMP	Calculates variable change across a contact surface
EXTRAP	Extrapolates entered values
INTERP	Interpolates entered values
DBURST	Calculates initial Riemann values at node where diaphragm is located
BONDRY	Calculates values to update left and right boundary
SRFLCT	Calculates new shock parameters when shock reflects off solid boundary
BBDRY	Alternate interpolating scheme near boundary
BORDER	Sets graphical borders
PLOT	Plots q, S, P, and ρ
EXACT	Plots exact ρ versus calculated ρ
LIST	Lists calculated values in files 8, 9, 10

open case, either constant pressure, or an adjustable pressure is specified. The latter case in effect extends the tube, and allows discontinuities to disappear out of the tube without creating expansion or compression waves. Exact values for velocities of expansion wave, shock, and contact surface, and density ratio are specified by the user in the input section.

Output is controlled by setting the variable GRAPHS to either 0, 1, or 2. A zero creates three files which contain data calculated at the time the call to subroutine LIST is made. A value of 1 causes plots of pressure, density, velocity, and entropy distributions to be created. When GRAPHS equals 2 a plot of the exact density distribution along with the calculated density distribution is made. Calls to output routines are determined by the parameter, SKIP, the number of time steps between calls, which is specified by the user.

Program termination can occur for several reasons. First, the program terminates when a maximum number of time steps, JSTOP, is reached. JSTOP is specified by the user. Second, the program terminates if, during execution, any of the following conditions are met:

1) The shock intersects with the contact surface
2) The pressure at any open boundary is higher than the pressure at the first node inside the boundary
3) The shock exits an open boundary.

In these cases a message is displayed on the terminal describing the reason for termination.

C. THE SUBROUTINE PROGRAMS

1. The "DBURST" Routine

Figure C.2 in Appendix C shows the flowchart for "DBURST". The assumption that a shock and contact surface are formed immediately causes oscillating numerical solutions that dampen within five to ten time steps. By mathematically "bursting" the diaphragm prior to time zero, a solution at the node where the diaphragm is located can be determined at time zero. The result is a more accurate representation of the wave structure and a dampening of a reduced transient solution within three time steps. "DBURST" calculates the Riemann variables that would exist behind a shock and contact surface that have moved an infinitesimal amount after the diaphragm burst, and assigns them to the node where the diaphragm was situated.

2. The "TIME" Routine

The "TIME" subroutine was not changed from that developed by Salacka [Ref. 2:p. 35]. The purpose is to compute the maximum allowable time step but ensuring that all characteristic trajectories over the time step are within one interval between nodes. The time step is determined from

$$DELT = H/ABS[Q+A] \tag{3.1}$$

at every node, and the minimum time step is used.

3. <u>The "TRAK" Routine</u>

This routine computes the new locations of the shock, contact surface, and head and tail of the expansion wave after the time step, DELT, calculated in "TIME." In Appendix C, Fig. C.3 shows the flowchart for the "TRAK" subroutine. The shock velocity, V_s, incoming Mach Number, w, and pressure, density, and sonic velocity ratios PR, DR, and AR respectively are determined. The equations for flow left and right are the same, except for DQ, the velocity gradient, which is different only by sign. Thus by proper bookkeeping the same equations are coded once and used in either direction. The initial direction of the shock and contact surface are set from code that interprets which side has the high pressure at time zero. The contact surface speed is determined after the shock speed is established. The situation at time zero is handled as in "DBURST" to ensure that the shock/contact surface interaction is properly modeled. Since there is a certain transient solution that decays after three time steps, tracking of the expansion wave is not initiated until then.

The head of the expansion wave travels at a velocity corresponding to q+A, and the tail with speed q-A. In the case of flow traveling to the left, this is reversed.

4. The "SWEEP" Routine

The "SWEEP" subroutine solves equation (2.5) and updates the values of variables at the nodes, including the boundary nodes. Appendix C, Fig. C.4 shows the "SWEEP" flowchart. Only those nodes which have been crossed by a discontinuity during the time step are updated by the "CORRCT" subroutine. Section II.B describes the theory coded into "SWEEP". The routine begins at the left boundary node and progresses to the right boundary node, one node at a time. Figure 3.1 illustrates the overall algorithm applying the QAZ1D method. At the boundary nodes, the

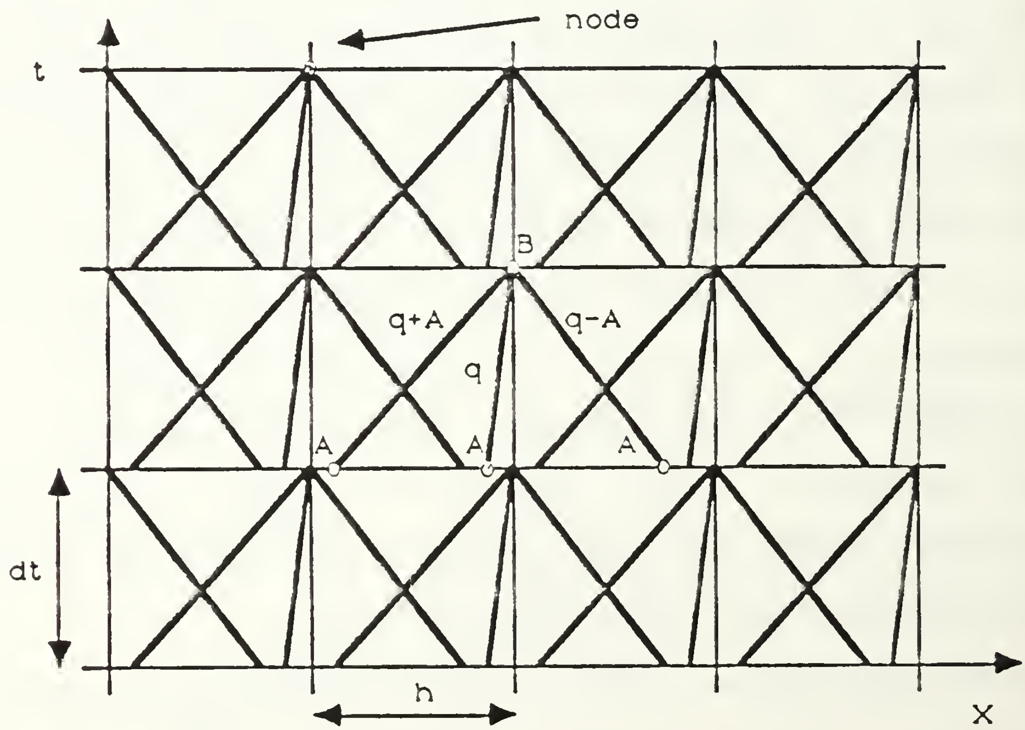


Figure 3.1 Overall Algorithm for QAZ1D Method

"BONDRY" subroutine is called and returns the updated values at those nodes. In between, the correct and most effective algorithm to update the node is determined. These algorithms are coded into eight different condition subroutines. These routines calculate the interpolated Riemann variables associated with the three characteristics ($q+A$, q, and $q-A$), plus the spatial derivatives at $\partial q/\partial s$ and $\partial A/\partial s$. This allows "SWEEP" to calculate $\Delta\overline{w}$ and $\overline{z}$. The integral of $\overline{z}$ is then determined, and the update for the variables of that node in $\partial\overline{w}$ are computed. $\partial\overline{w}$ is stored until the entire mesh has been swept. Then the variable arrays ($\overline{w}$) are updated to the next time interval.

To choose the proper condition routine, the following information about conditions in the interval on either side of the node are expressed in two 3 digit integer variables, SHOCK and CNTACT. The value of these variables provide a code for the following information:

SHOCK--The existence of a shock within an interval, and, if so, the direction of travel, and if the shock will cross the node in front of it within the current time step.

CNTACT-The existence of a contact surface within an interval, and, if so, the direction of travel, and if the contact surface will cross the node in front of it within the current time step.

Figure 3.2 illustrates the regions around a node. Table V lists the values that SHOCK and CNTACT may have, and their meaning. Figures 3.3, 3.4, and 3.5 illustrate examples of what different SHOCK and CNTACT values mean. At each node

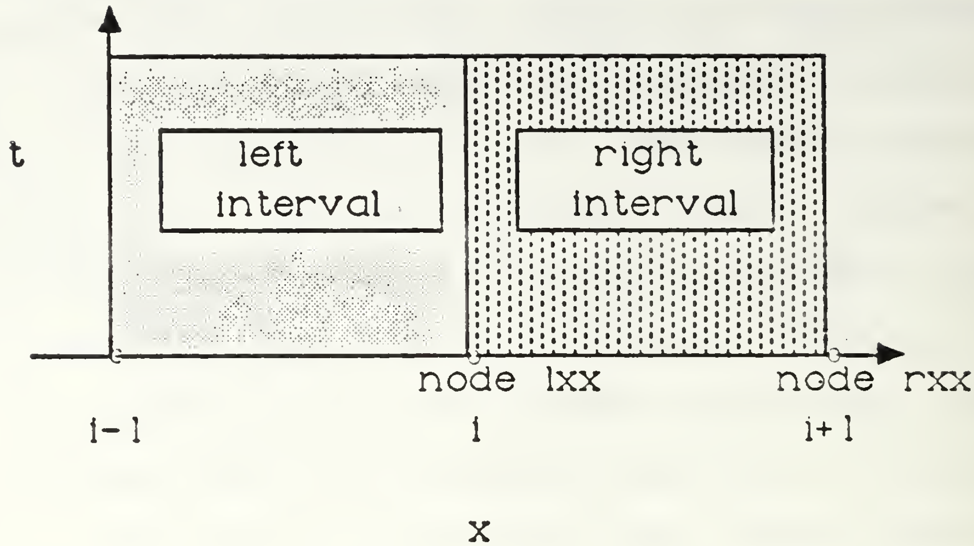


Figure 3.2 Interval and Node Description used in "SWEEP"

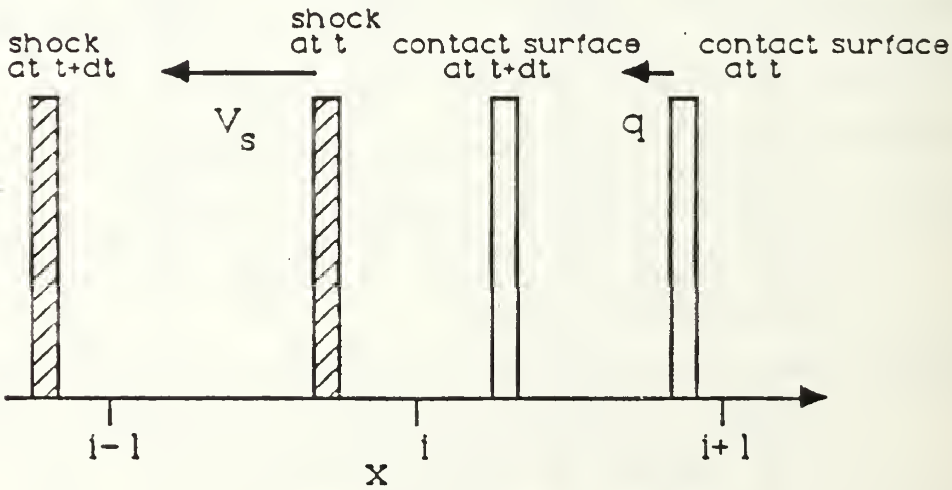


Figure 3.3 Schematic of SHOCK = 331 and CNTACT = 232

TABLE V

VALUES OF PARAMETERS SHOCK AND CNTACT

SHOCK

Value	Located in interval on	Direction of travel	Crosses node in path within time step
100	No Shock	N/A	N/A
321	Left	Right	Yes
322	Left	Right	No
331	Left	Left	Yes
332	Left	Left	No
221	Right	Right	Yes
222	Right	Right	No
231	Right	Left	Yes
232	Right	Left	No

CNTACT

Value	Located in interval on	Direction of travel	Crosses node in path within time step
100	No contact surface	N/A	N/A
321	Left	Right	Yes
322	Left	Right	No
331	Left	Left	Yes
332	Left	Left	No
221	Right	Right	Yes
222	Right	Right	No
231	Right	Left	Yes
232	Right	Left	No

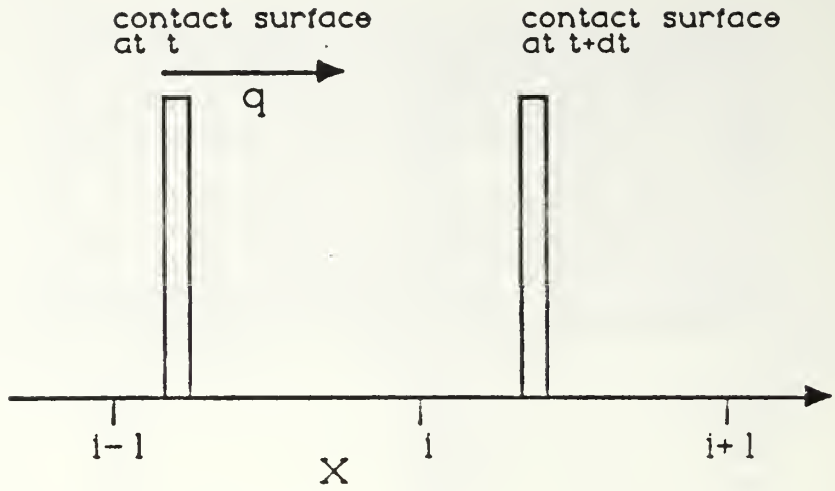


Figure 3.4 Schematic of SHOCK = 100 and CONTACT = 321

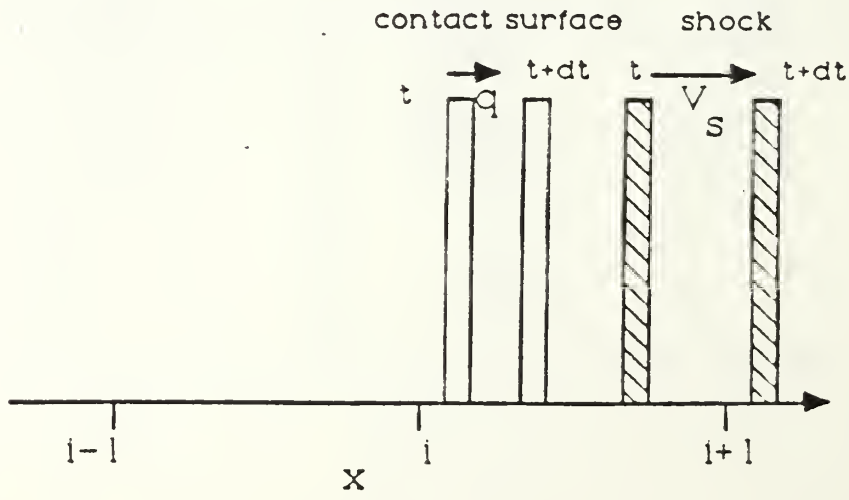


Figure 3.5 Schematic of SHOCK = 221 and CNTACT = 222

the code examines the shock and contact surface condition, along with relative location of the shock to a contact surface if both exist near the node, if the flow is supersonic or subsonic at the node, and the direction the flow is traveling. The appropriate algorithm is determined and called in the form of a condition subroutine.

The "SWEEP" routine is designed to handle all possible shock and contact surface combinations. Because the code does not solve the situation where the shock and contact surface intersect, as illustrated in Fig. 3.6, then any combinations after the shock and contact surface have crossed call condition routines that currently contain only messages. The code was intentionally prepared to be easily extended to treat the intersection of the shock and the contact surface, and beyond.

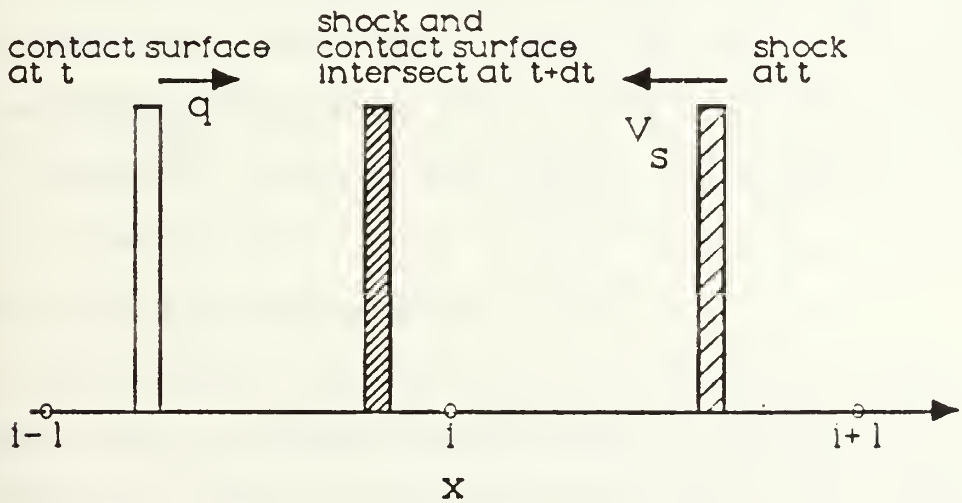


Figure 3.6 Schematic of SHOCK = 231 and CNTACT = 322

5. The "Condition" Subroutines

The ten subroutines that are called by "SWEEP" to calculate the interpolated values of the extended Riemann variables, R and Q, plus S, $\partial q/\partial s$, and $\partial A/\partial s$ are COND1, COND2, COND3, COND4, COND5, COND6, COND7, COND7S, COND7N, and COND8. The procedure used in COND1, COND2, COND3, and COND5 is that of Salacka [Ref. 2:pp. 36-37] modified to account for contact surfaces. Appendix C, Fig. C.5 is a general flowchart for these four routines. Essentially, an initial estimate is made of the characteristic slopes, λ, by enforcing the principle of domain of dependence. The assumption is made that the slopes are linear, and that the characteristics pass through point B, as defined in Fig. 3.7. Then q and A are computed at point A, which in turn yields a second estimate of the characteristic slope, λ.

The two slopes for each characteristic are compared. If they are not equal to within an acceptable error, the new estimate of λ is used to repeat the process until convergence is achieved. All three characteristics are handled simultaneously. The value of

$$\Delta s = \lambda \cdot \partial t \tag{3.2}$$

is then determined for each characteristic $q+A$, q, and $q-A$. This allows linear interpolation of Q, R, and S at point A. The spatial derivatives, $\partial Q/\partial s$ and $\partial A/\partial s$ are computed from

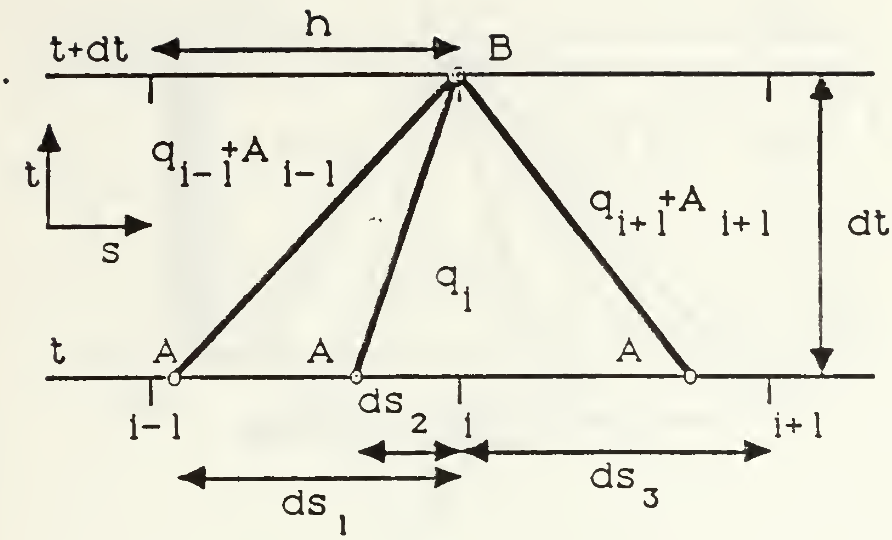


Figure 3.7 Computational Method for COND1 Routine

the values of q and A used in the estimate of the initial characteristic slopes.

The "COND1" subroutine is used when there are no contact surfaces or shocks within the interval on either side of the node, and flow is subsonic. The q+A characteristic uses forward differencing, while the q-A characteristic uses a backward differencing scheme to keep the initial domain of dependence of the numerical scheme outside the physical domain of dependence.

The "COND2" subroutine is a backward differencing algorithm used in situations when one or more discontinuities are in the right interval, or if flow is supersonic to the right. Fig. 3.8 illustrates the computational method

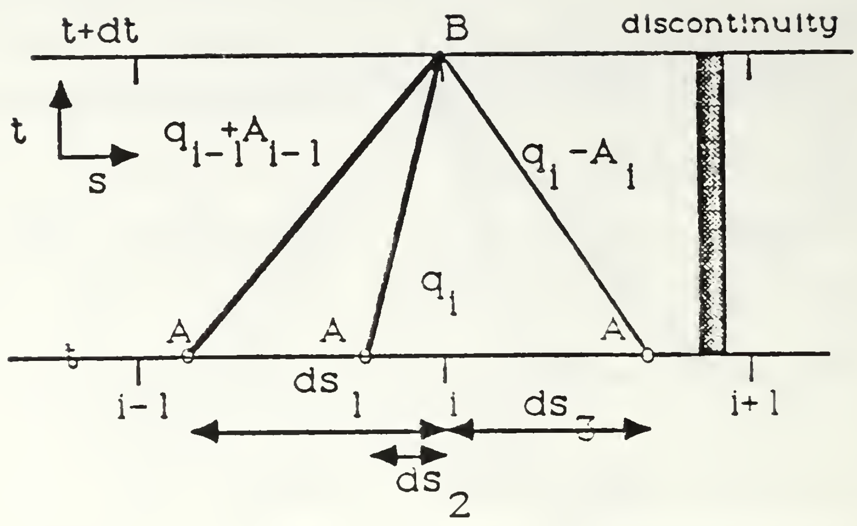


Figure 3.8 Computational Method for COND2 Routine

for "COND2". The characteristic slopes of q+A and q are handled as in "COND1", but the q-A characteristic slope is determined by a backward rather than a forward scheme. To interpolate between node i and i+1 would be incorrect because of the discontinuity in the values between them. The value of parameters in the right interval are interpolated from values in the left interval by assuming the derivatives do not change between the intervals up to the discontinuities (i.e., a shock and/or a contact surface).

The "COND3" subroutine is a forward differencing algorithm used in situations when one or more discontinuities are in the left interval, or if flow is supersonic traveling to the left. Figure 3.9 shows the computational

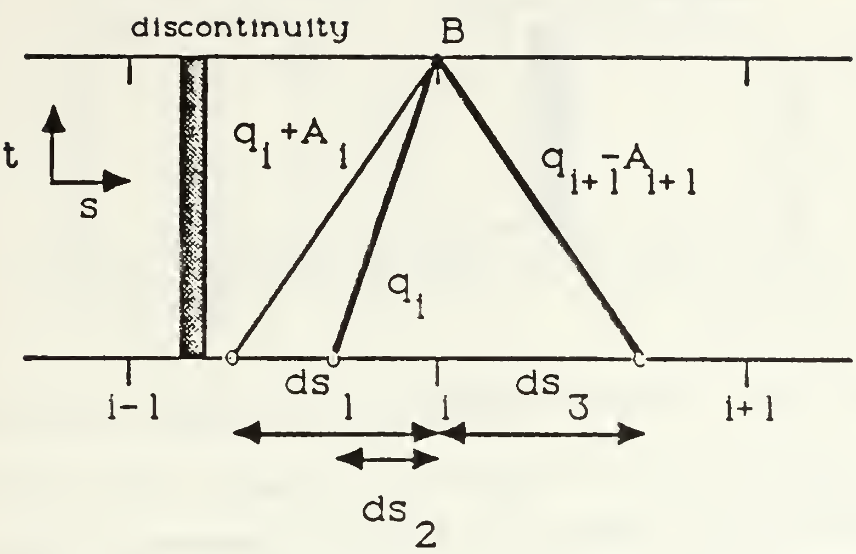


Figure 3.9 Computational Method for COND3 Routine

method for "COND3". The method is similar to that in COND2 but the q+A characteristic slope has to be interpolated from values of node i instead of node i-1 using a forward difference scheme. The characteristic slope for q-A and q are estimated by forward difference schemes using values of node i+1 and i respectively.

For conditions, such as that illustrated in Fig. 3.10, where a discontinuity is in the interval opposite the direction of supersonic flow then COND5 subroutine is called. A mix of COND2 and COND3 is used. For flow to the right the method of COND2 is implemented, while flow to the left is the same as for COND3.

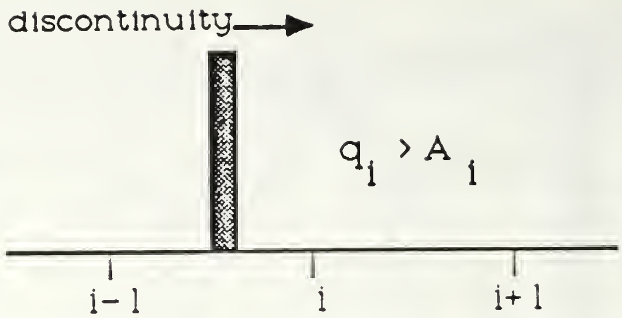


Figure 3.10 Example Condition for "COND5" Routine

Subroutine "COND4" is called whenever a node is to be crossed by a discontinuity from either direction. The flowchart for "COND4" is shown in Appendix C, Fig. C.6. Two integer vector arrays, LNODE and RNODE, are used to store the following information on the node being crossed.

A) the node number, I

B) the value of SHOCK

C) the value of CNTACT

D) the current value of the integer time step, J

The information is used in "CORRCT" to update the node to the next time level. In the current code the maximum number of nodes that can be crossed during any time step is two. Figure 3.11 shows an example of this situation, and defines the assignment of values. Because the nodes are swept from left to right, the left-most node is assigned to LNODE, while RNODE applies to any node crossed to the right of the LNODE node. The values of $\Delta\overline{w}$ are set to zero, so

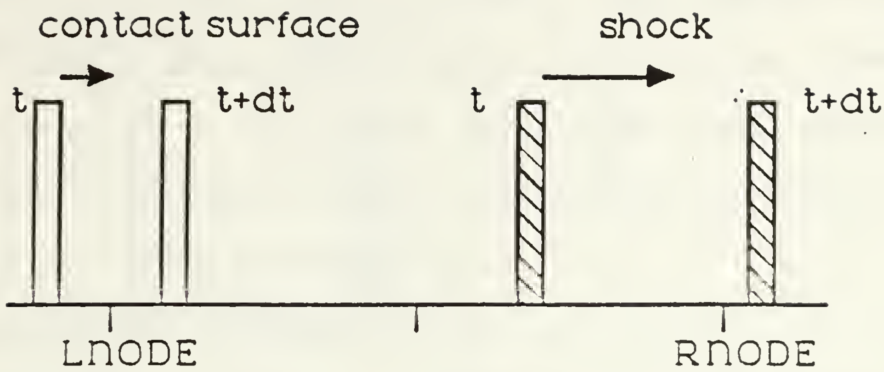


Figure 3.11 Example of "COND4" Routine Situation

that these nodes are skipped when updating occurs in "SWEEP". The main program interprets the value of J in LNODE and RNODE to determine if zero, one, or two nodes need to be corrected in "CORRCT". If zero nodes are crossed during a time step then "CORRCT" is not called.

The "COND6" and "COND8" subroutines are designed for future extensions of the entire code. Until the Riemann problem illustrated in Fig. 3.6 is coded, then any situation after the shock and contact surface intersect cannot be computed. However, "SWEEP" is already coded to call "COND6" for the situation illustrated in Fig. 3.12, or its mirror image. "COND8" would be called for all the other situations after the shock and contact surface interaction has taken place. Currently, both subroutines output messages on the screen describing the situation, and set HALT equal to 1 to terminate the program.

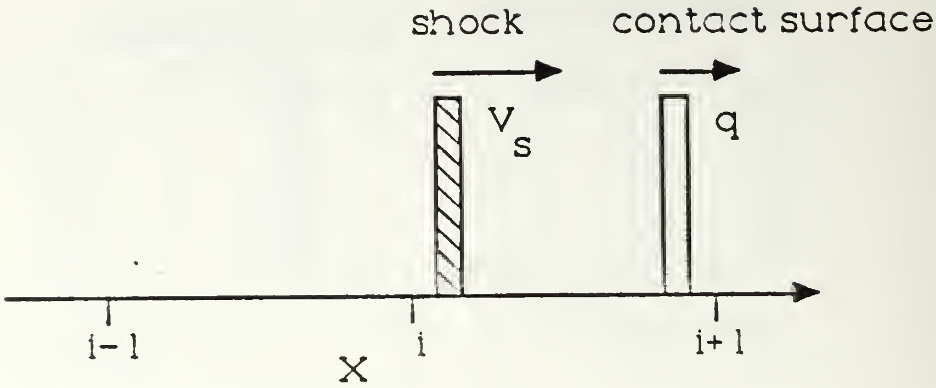


Figure 3.12 Example of "COND6" Situation

The "COND7", "COND7N", and "COND7S" subroutines are all related to each other. They are intended to contain code that would facilitate the shock and contact surface intersection and solve the resulting Riemann problem. The "COND7" routine is called when the contact surface and shock moving in opposite directions would cross in a time step within an interval. The routine would calculate when and where within the time and space domain the intersection would occur based on the known velocity of each discontinuity and their current locations. This new time could then be used to rerun "SWEEP" with the shock and contact surface exactly on top of each other at the end of the new time step. The "COND7N" routine is for the same situation, but when a node is crossed by either discontinuity during the same time step. The same procedure is done, with the requirement to ensure the node crossed is properly updated

by "CORRCT" during the new time step. The "COND7S" routine is called when a shock and contact surface are located at the same point at a time other than zero. The subroutine would contain the code to solve the Riemann problem set up by "COND7N" or "COND7" routines.

6. <u>The "CORRCT" Routine</u>

Appendix C, Fig. C.7 shows the flowchart for "CORRCT". The "CORRCT" routine corrects nodes that have been crossed by a discontinuity (i.e., a shock or a contact surface) or are straddled by both. The routine takes the information from LNODE and RNODE arrays to determine which node or nodes are to be updated, and if there is any shock and contact surface interaction at the nodes. Then, using subroutines "DELTAX", "SKJUMP", "CSJUMP", "EXTRAP", "INTERP", and "BBDRY" the concepts from Section II are imposed to calculate the jump in the parameters R, Q, S, A and q at the node in question to the new time level.

7. <u>The "BONDRY" Routine</u>

The "BONDRY" routine computes the new values of the parameters Q, R, and S at the boundary nodes for time step t. The concepts of Section II for an open or closed boundary are coded as shown in the flowchart of "BONDRY" given in Appendix C, Fig. C.8. The routine uses the "COND1", "COND2", or "COND3" routines to calculate the correct w, and then solves for w the same as in "SWEEP". If a shock crosses an open boundary node during a time step,

then "SKJUMP" routine is used to calculate the values behind the shock at the node.

8. The "SRFLCT" Routine

The "SRFLCT" subroutine is called to calculate the values of Q, R, S, q and A at the boundary node when a shock reflects from a solid boundary. Appendix C, Fig. C.9 shows the flowchart of "SRFLCT" which codes the analysis in Section II.D.2 for closed boundary shock reflection. The time for the shock to reach the solid boundary, ∂t_W, is computed. Then the excess time in the time step, ∂t, is given by

$$\partial t_{ex} = \partial t - \partial t_W \tag{3.3}$$

After the new shock velocity, V_s, is calculated, then the new location of the reflected shock is known from multiplying V_s by ∂t_{ex} and adding or subtracting from the boundary location.

9. The "DELTAX" Routine

The "DELTAX" subroutine is used in "CORRCT" and "BONDRY" to calculate the location within an interval of a discontinuity in terms of x. Figure 3.13 defines the various displacements which are calculated.

10. The "SKJUMP" Routine

The "SKJUMP" subroutine computes the conditions downstream of a stationary normal shock as described in

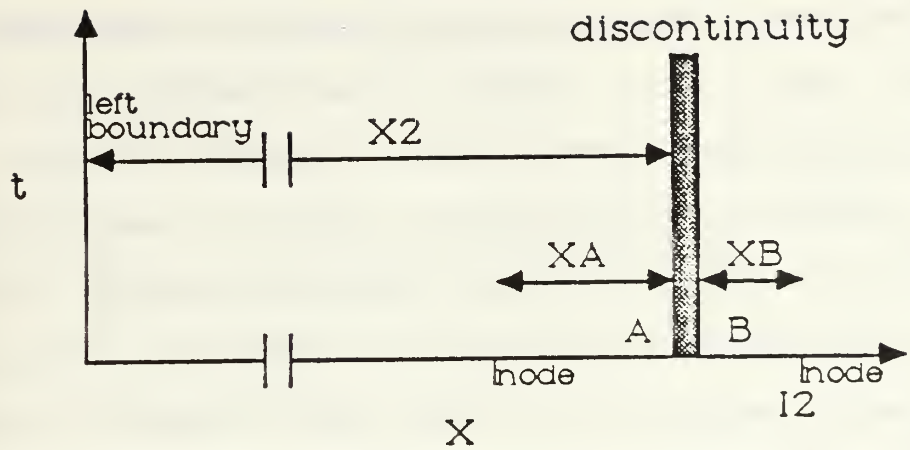


Figure 3.13 Discontinuity Location within an Interval

Section II.D.2. The "CORRCT", "TRAK", and "BONDRY" subroutines call "SKJUMP" when required, to obtain the speed of sound (A), entropy (S), and velocity (q) behind the shock.

11. The "CSJUMP" Routine

The "CSJUMP" subroutine computes the velocity and speed of sound change across a contract surface as described in Section II.D.3. The "CORRCT" and "TRAK" routines call "CSJUMP".

12. Numerical Support Routines

The "EXTRAP", "INTERP", and "BBDRY" routines are used by "CORRCT", "BONDRY", "TRAK", and "DBURST" to extrapolate or interpolate data to the face of a discontinuity or point. In particular, "BBDRY" is an interpolation routine used within an interval near a boundary.

13. The Output Routines

The four output subroutines used are those developed by Salacka [Ref. 2:pp. 39-40]. The "BORDER" and "PLOT" routines output plots of pressure, velocity, density, and modified entropy distributions versus a non-dimensionalized tube length at preset time intervals. "EXACT" creates a plot of the exact density distribution at selected points and compares it with the computed density distribution for a selected time step. The "LIST" routine outputs to files 8, 9, and 10 tabular listing of computed data. The files are sent to the user's permanent storage disk.

A value of GRAPHS equal to one causes "BORDER" to be called in the main program. This defines plot axis, labels, and headings. "PLOT" is then called once at time zero, and at every time step set by SKIP.

If GRAPHS is set equal to two, then "EXACT" is called every SKIP time steps to plot six exact density values and the computed density distribution. The six exact points are:

A) The two boundary points

B) The head and tail of the rarefaction wave

C) A point just behind the contact surface

D) A point just behind the shock.

The location of the exact values is based on elapsed time and known exact values for wave velocities entered with the initial conditions for the problem.

"LIST" routine is called every SKIP time steps when GRAPHS is equal to zero. File 9 contains a tabular listing of the Riemann variables, modified entropy, pressure, density and velocity distributions, elapsed time, discontinuity velocity and location. File 10 is a tabular listing of the location of the shock, contact surface, and the head and tail of the expansion wave computed at every SKIP time steps. File 8 contains the exact values of the location of the shock, contact surface, and the head and tail of the expansion wave.

IV. RESULTS

Four test cases were run using the E1DV2 code. The purpose was to verify the ability of the code to determine the unsteady flow process and correctly simulate varying boundary conditions and flow directions in the Riemann shock tube problem. The shock tube is illustrated in Fig. 4.1 with the high pressure on the left. The tube is divided into two sections by a diaphragm. When the diaphragm is burst, the pressure equalizes through a shock wave traveling into the expansion chamber, and an expansion or rarefaction

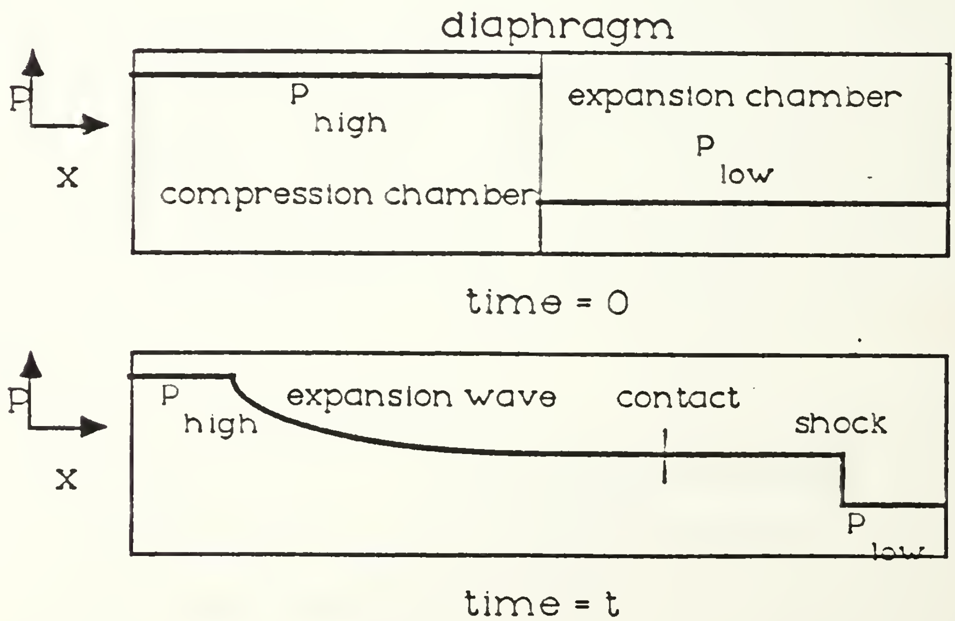


Figure 4.1 Shock Tube at t = 0 and t = t

wave traveling into the compression chamber. A contact discontinuity is behind the shock wave, traveling at the particle velocity [Ref. 6:p. 30].

A. TEST CASE 1

Test Case 1 was designed to demonstrate shock reflection and expansion wave reflection at solid boundaries. Test Case 1 had the following initial and boundary conditions:

1) Pressure and density ratios equal to five, with high pressure on the left
2) Temperature ratio equal to unity
3) Both boundaries were closed
4) Diaphragm was located at x = 0.5
5) Computational mesh had 101 nodes
6) Maximum time step, JSTOP, = 109, with SKIP = 18.

Plots of the results obtained for the pressure, density, velocity, and modified entropy distributions are shown in Fig. 4.2 and Fig. 4.3. Fig. 4.2 shows the computation up to time step 55, while Fig. 4.3 takes the computation from time step 56 to termination. The output of "EXACT", a plot of the exact density compared with the computed density distribution is shown in Fig. 4.4. Fig. 4.5 illustrates the spatial location versus time for exact and computed shock, contact surface, and head and tail of the expansion wave. Data for Fig. 4.5 were taken from file 10 and file 8. Fig. 4.6 shows plots of pressure, density, modified entropy, and velocity distributions for the same initial and boundary

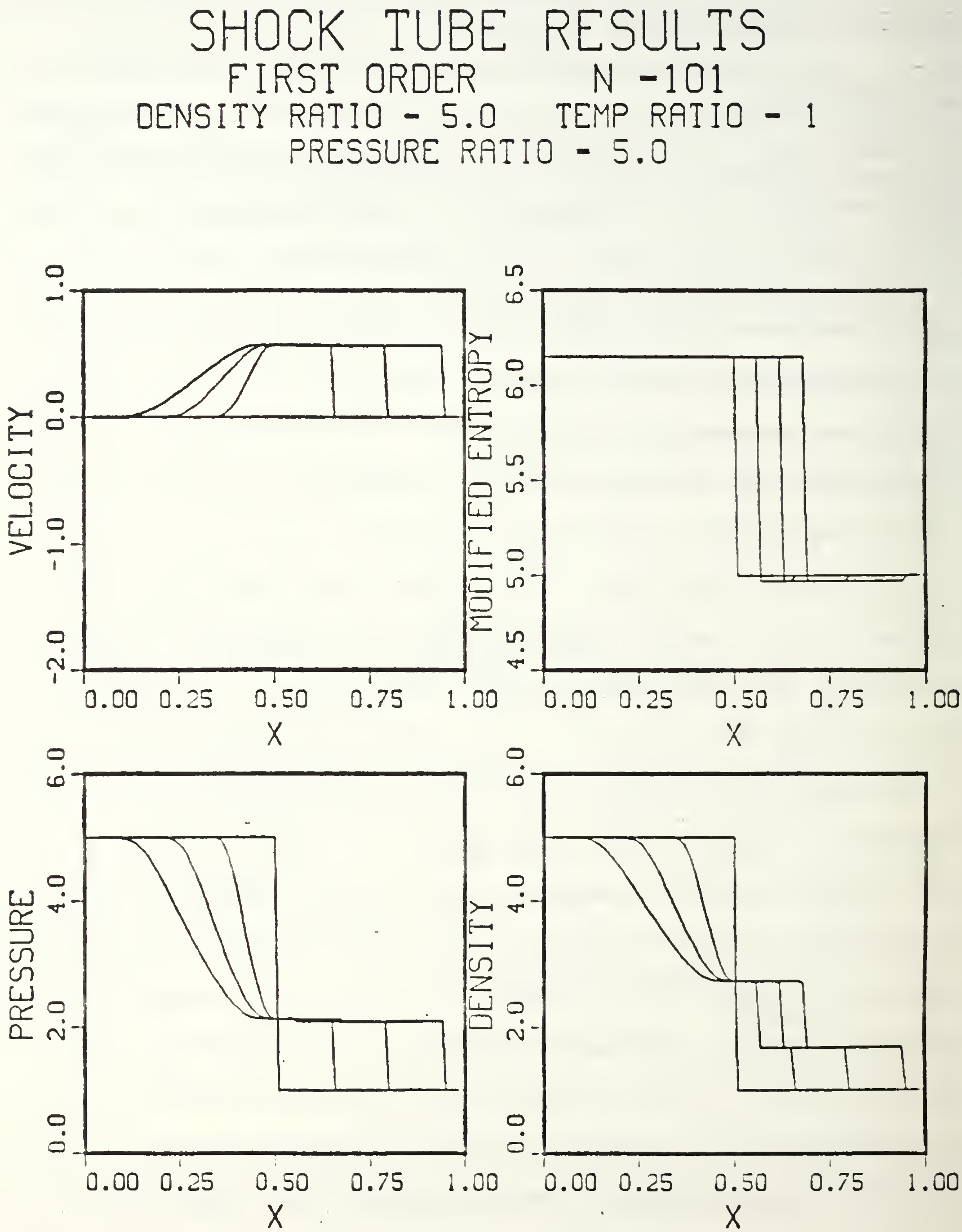


Figure 4.2 Test Case 1 (J = 1 to J = 55)

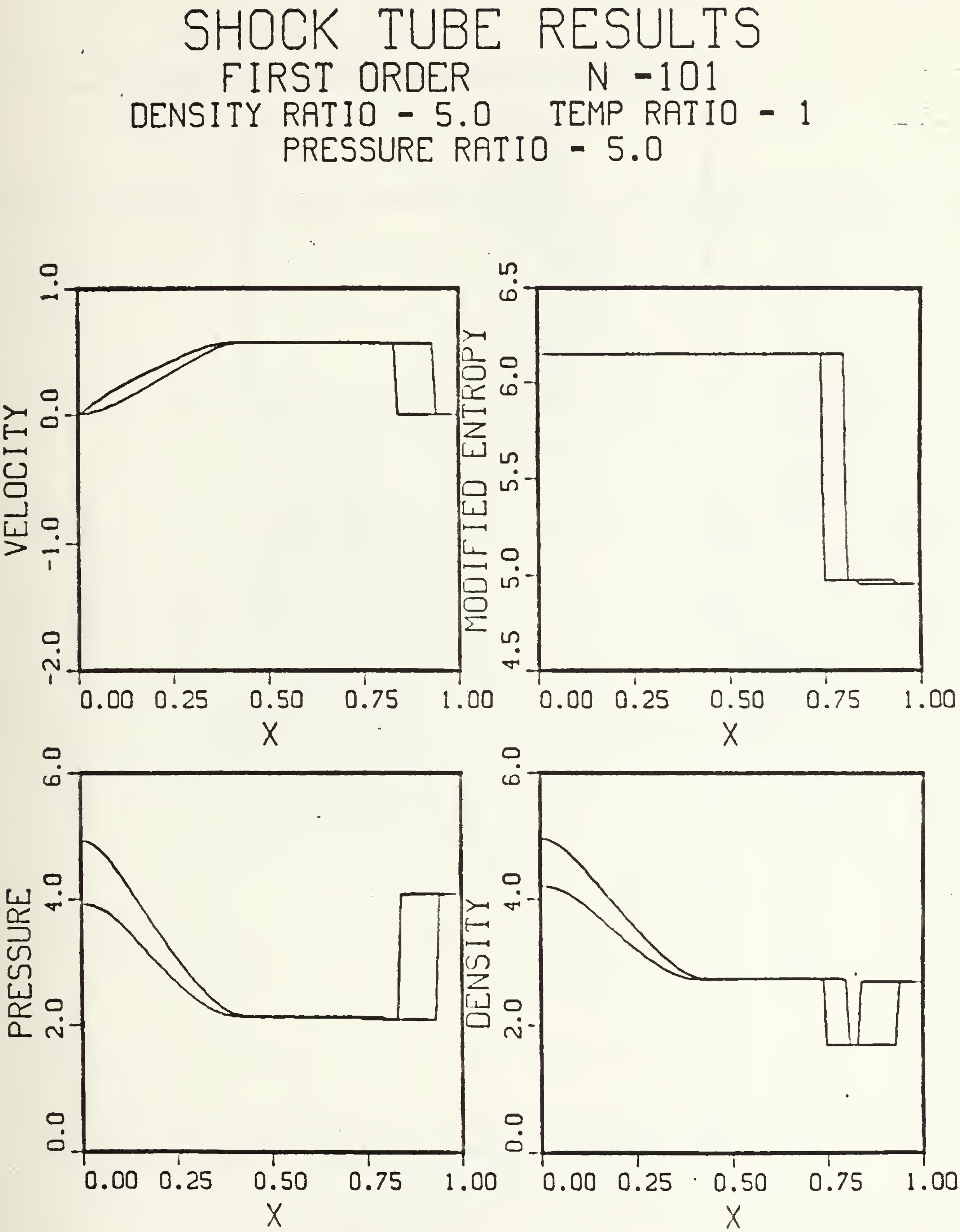


Figure 4.3 Test Case 1 (J = 56 to J = 109)

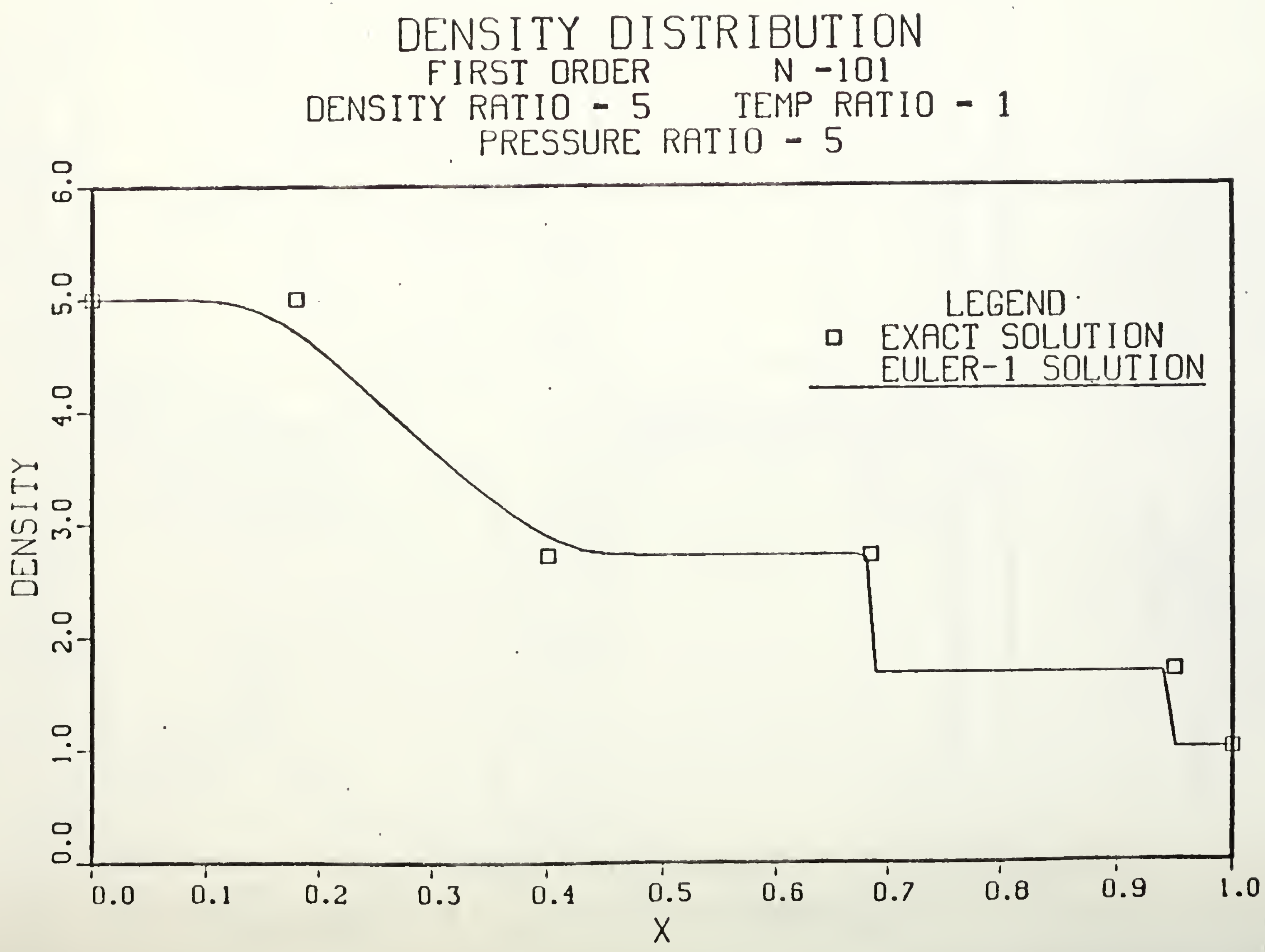


Figure 4.4 Exact and Computed Density Distributions

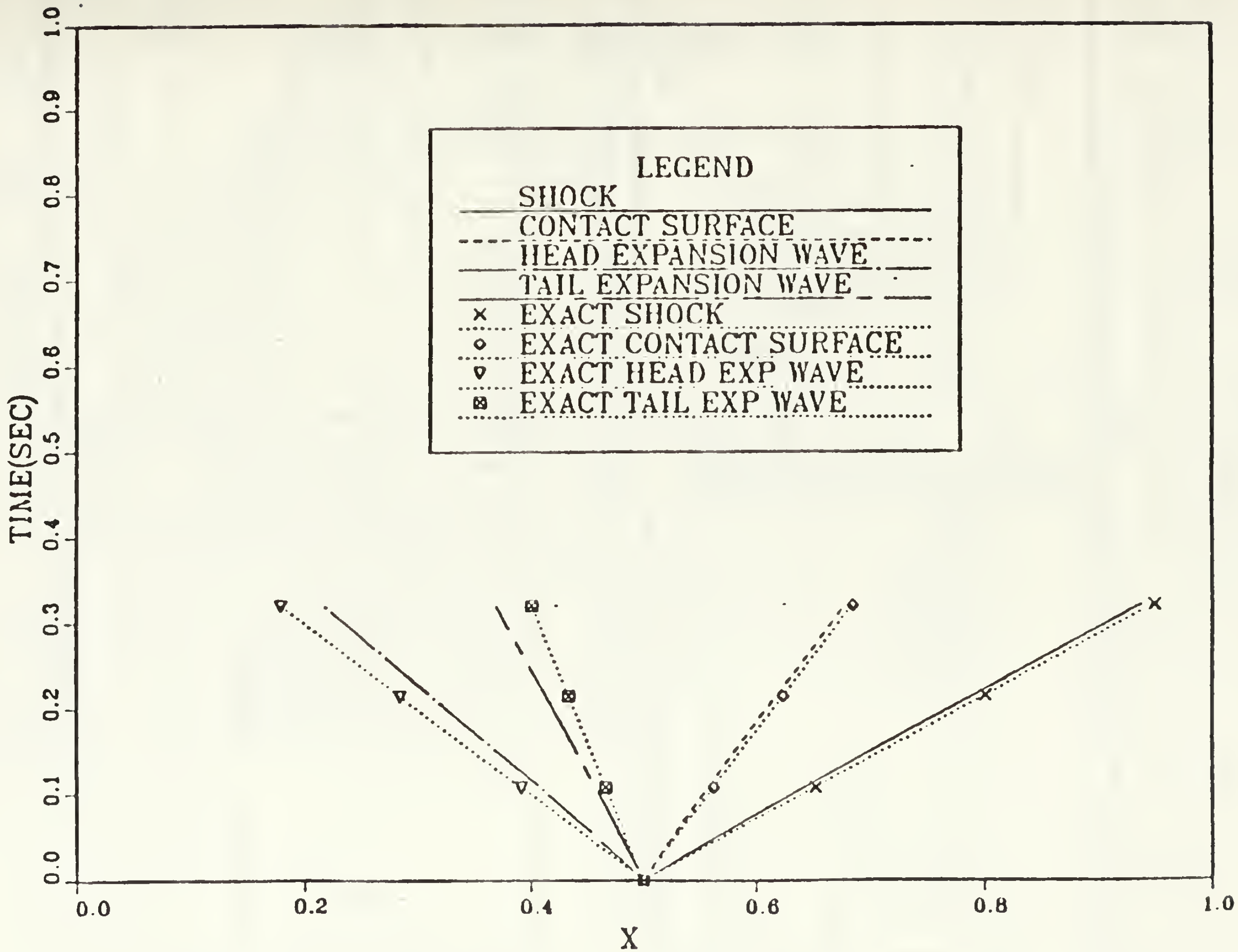


Figure 4.5 Location of Discontinuities versus Time (Test Case 1)

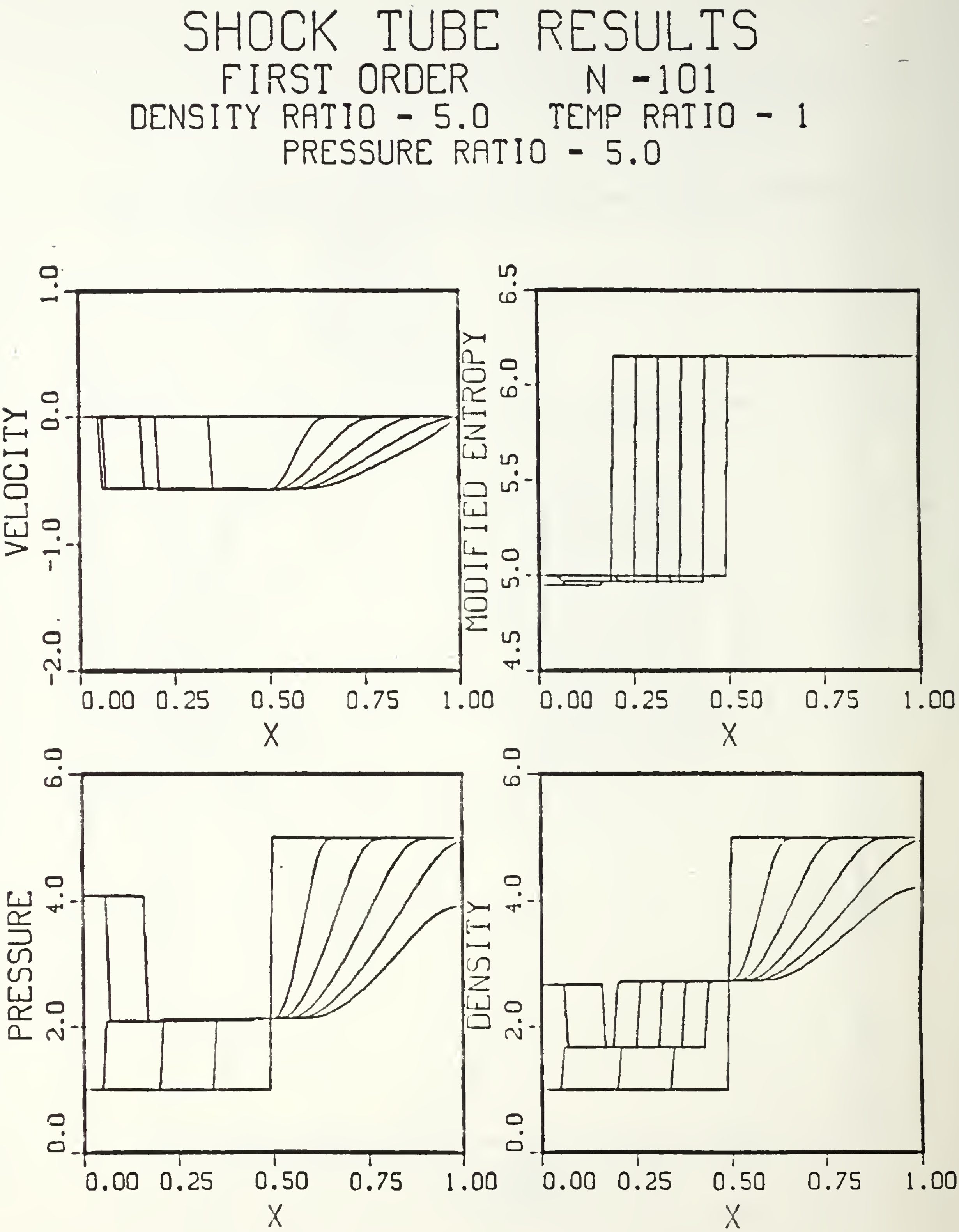


Figure 4.6 Test Case 1, High Pressure on Right

conditions but with high pressure on the right. Figure 4.6 includes data corresponding to those in both Fig. 4.2 and Fig. 4.3 for the high pressure initially on the left.

B. TEST CASE 2

Test Case 2 was to demonstrate an open boundary, constant pressure, expansion wave interaction. The test case was run with the following initial and boundary conditions:

1) Pressure and density ratios equal to 5.0, with high pressure on the left

2) Temperature ratio equal to unity

3) The left boundary was open, with a constant pressure ratio across the boundary of 4/5

4) The right boundary was open, with an adjustable pressure ratio that in effect extended the length of the tube

5) The diaphragm was located at x = 0.74

6) Computational mesh had 51 nodes

7) Maximum time step, JSTOP, = 37, SKIP = 9.

Plots of the results for the pressure, density, velocity, and modified entropy distributions are shown in Fig. 4.7. The test case was rerun with high pressure on the right. Figure 4.8 shows the results for pressure, density, and temperature ratios the same. The diaphragm was then at x = 0.26. Boundary conditions were reversed. The same computational mesh, and output control were used.

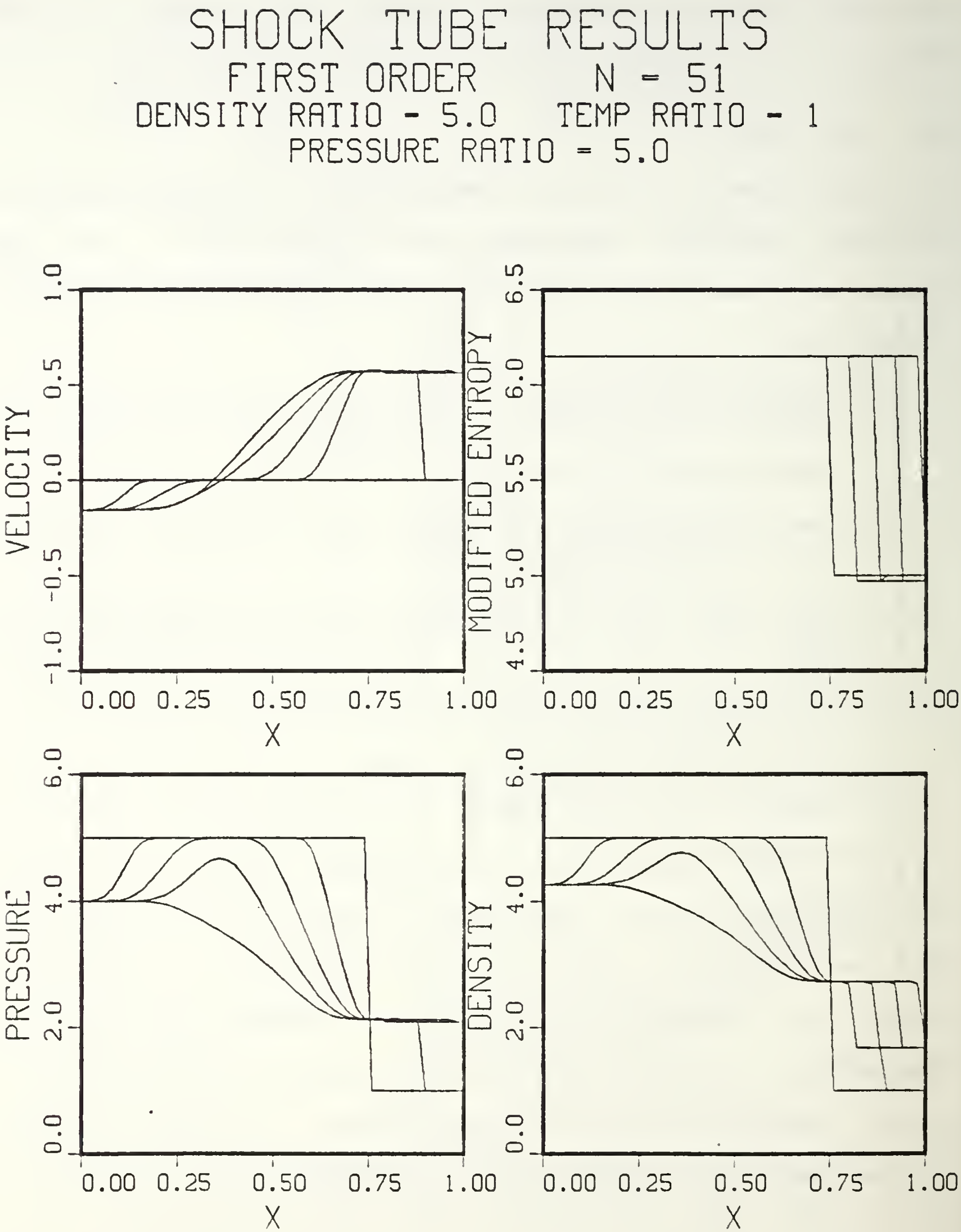


Figure 4.7 Test Case 2, High Pressure on Left

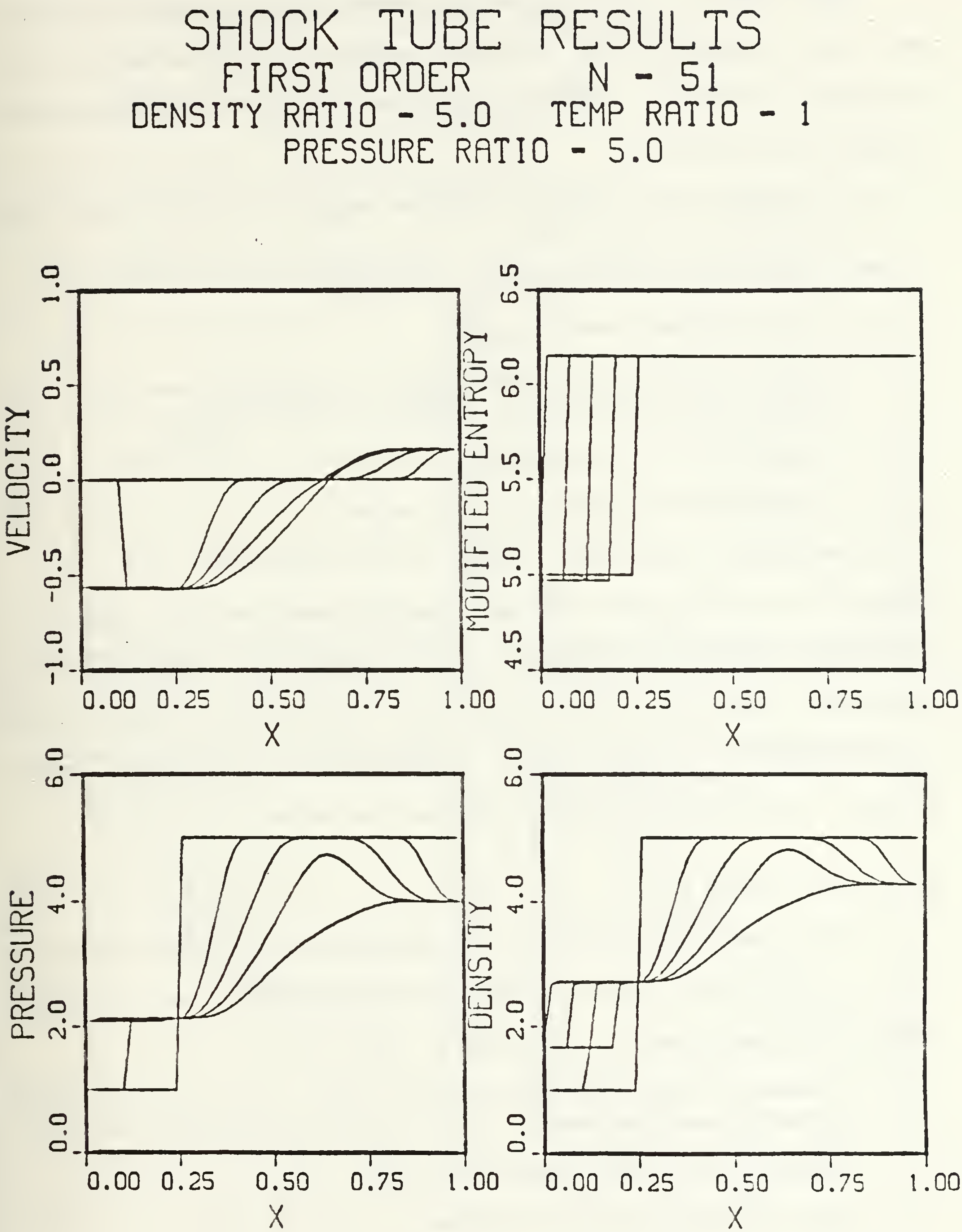


Figure 4.8 Test Case 2, High Pressure on Right

C. TEST CASE 3

Test Case 3 was designed to demonstrate a shock exiting an open boundary with constant pressure maintained at the boundary itself. The following initial and boundary conditions were set:

1) Pressure and density ratios equal to 5.5, with high pressure on the left

2) Temperature ratio equal to unity

3) Left boundary was closed

4) Right boundary was open with a constant pressure ratio across the boundary of unity

5) Diaphragm was located at x = 0.5

6) Computational mesh had 101 nodes

7) Maximum time step, JSTOP, = 73, with SKIP = 18.

Figure 4.9 shows plots of the results obtained for pressure, density, modified entropy, and velocity distributions. Similarly, Fig. 4.10 shows results obtained by putting the high pressure on the right, reversing the boundary conditions, and holding everything else the same.

D. TEST CASE 4

Test Case 4 was designed to demonstrate a lower pressure ratio, with shock wave reflection. The initial and boundary conditions were set as follows:

1) Pressure and density ratios equal to 3.2, with high pressure on the left

2) Temperature ratio equal to unity

3) Both boundaries were closed

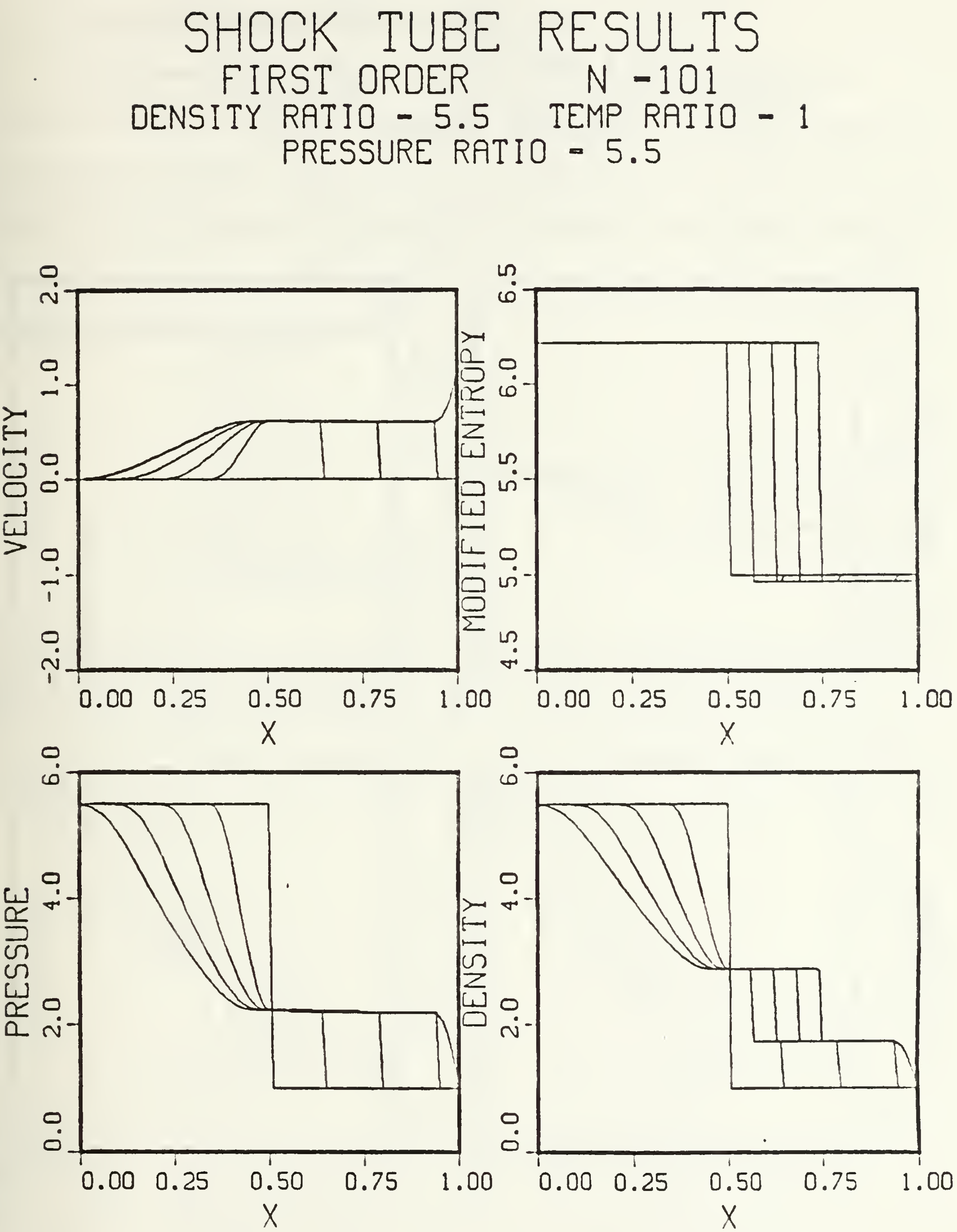


Figure 4.9 Test Case 3, High Pressure on Left

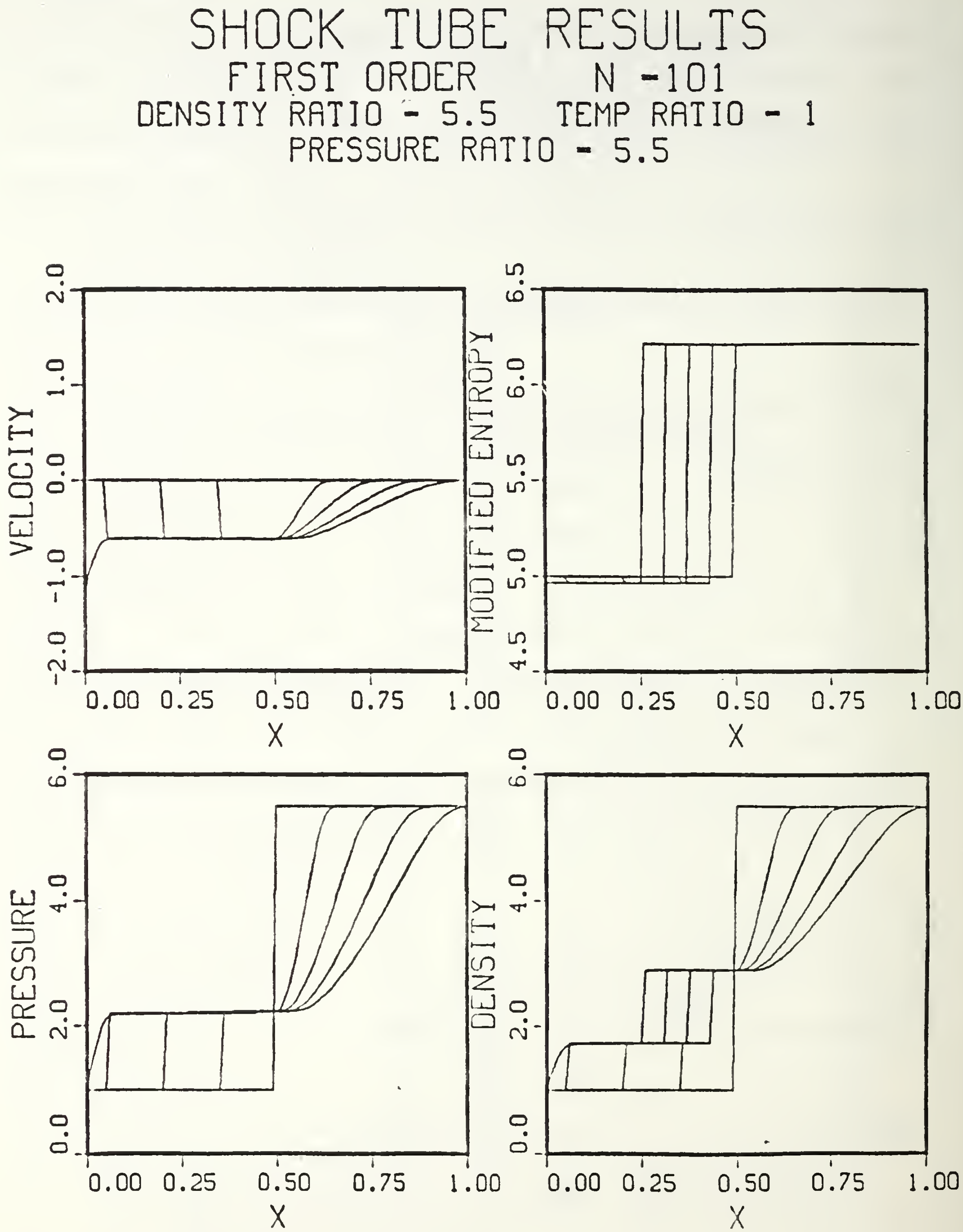


Figure 4.10 Test Case 3, High Pressure on Right

4) Diaphragm was located at x = 0.5

5) Computational mesh had 901 nodes

6) Maximum time step, JSTOP, = 601, with SKIP = 150.

Plots of the results obtained for the pressure, density, modified entropy, and velocity distributions are given in Fig. 4.11. Figure 4.12 shows the results obtained with the high pressure to the right, SKIP = 200, and holding all other parameters constant.

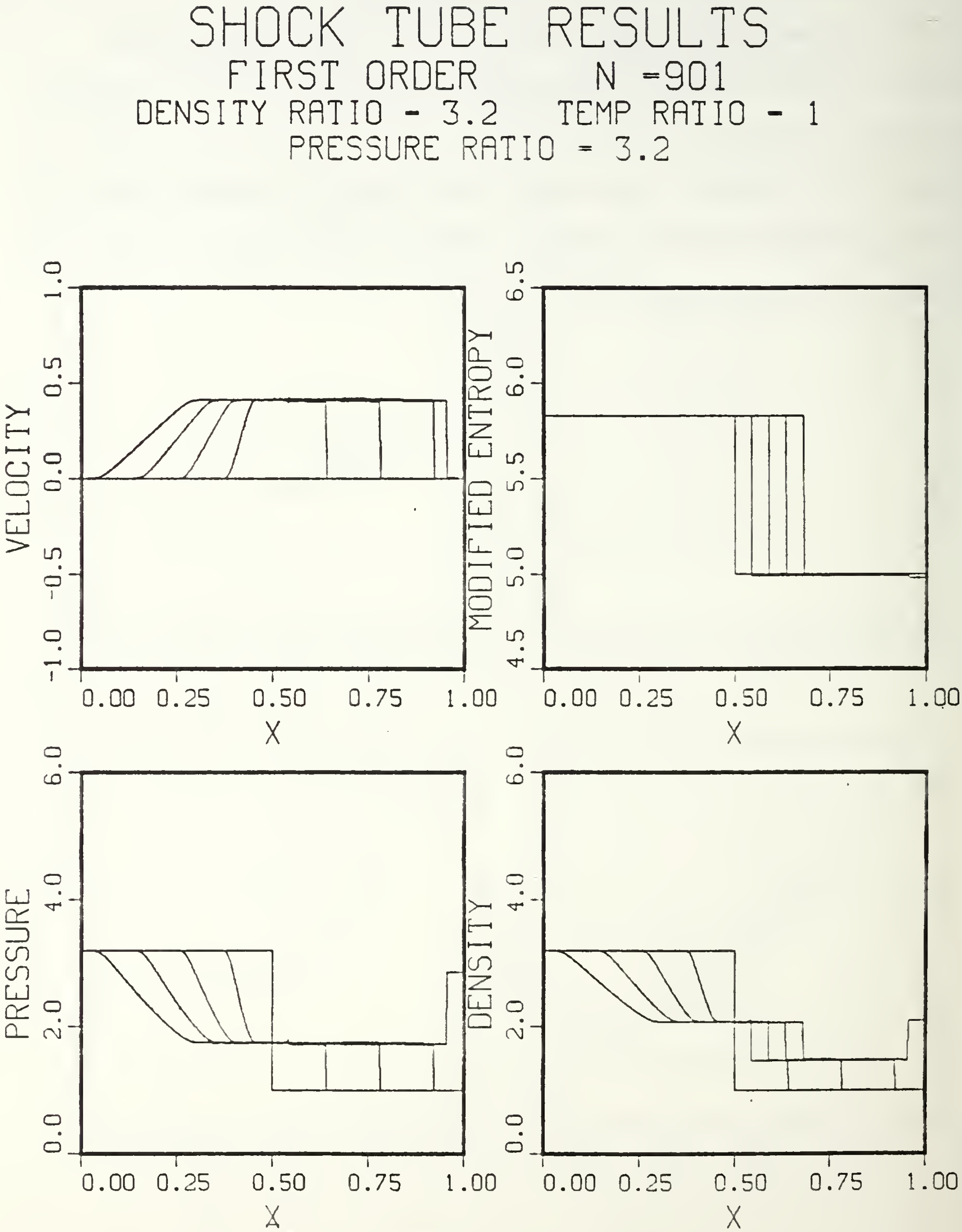


Figure 4.11 Test Case 4, High Pressure on Left

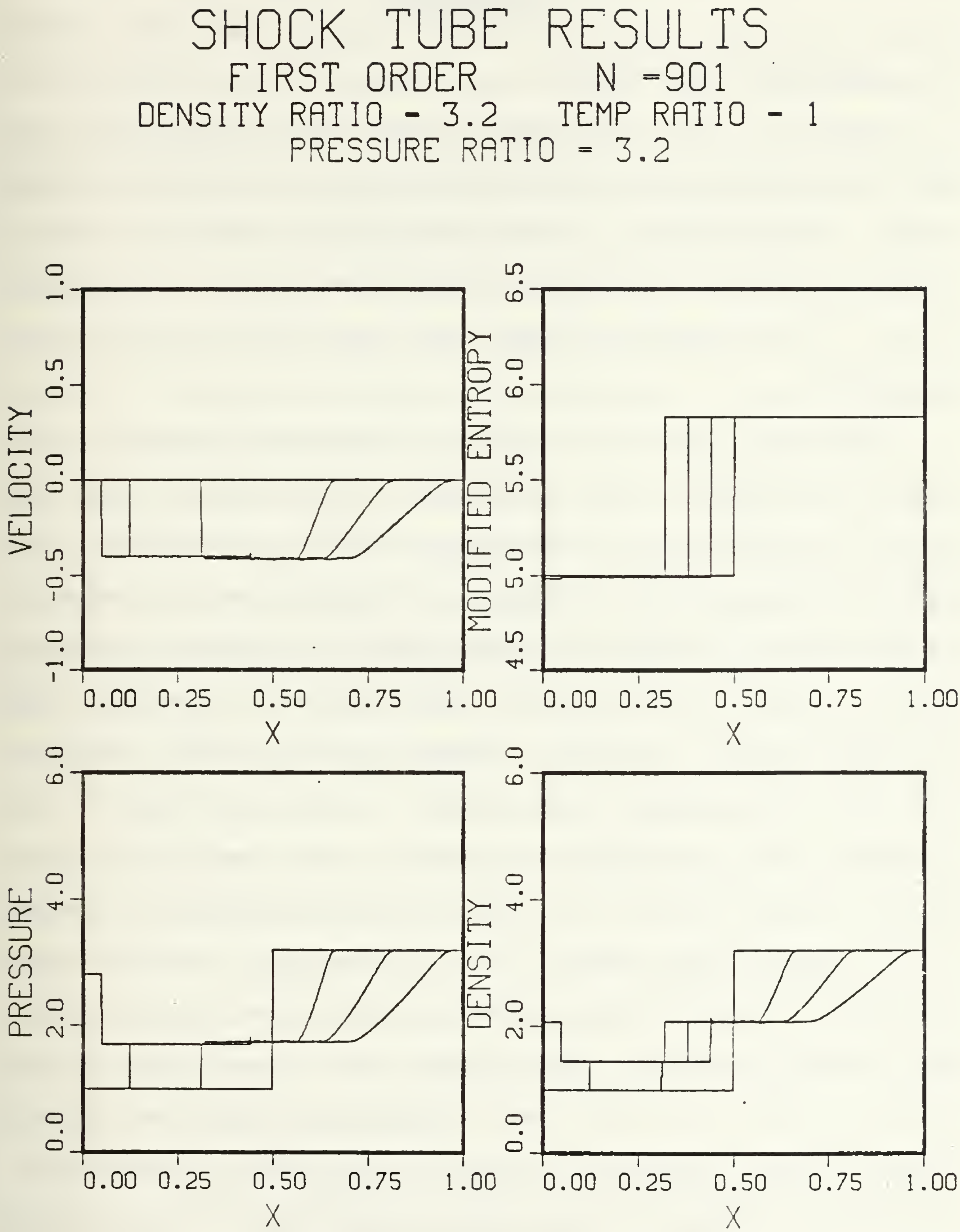


Figure 4.12 Test Case 4, High Pressure on Right

V. DISCUSSION

A. RESULTS OF TEST CASES

Overall, in the four specific test cases which were run, the expected qualitative flow behavior was produced by the code. Quantitatively, either experimental data or further exact solutions are needed to fully validate the computations. The results of each test case will be discussed separately.

Test Case 1 results in Fig. 4.2 show the formation of a well-defined shock wave traveling to the right, with the contact surface crisply defined following along behind. The entropy drops sharply across the shock and remains constant to the contact surface and then jumps to a steady, constant value to the left boundary as expected. Pressure and velocity remain constant through the contact surface. Velocity is positive since flow is to the right. The expansion wave is smeared from head to tail over the correct range. Figure 4.3 is a continuation of the flow problem with a sequential display of the results. The entropy continues to drop as the reflected shock travels back toward the contact surface. Notice that the velocity behind the shock is zero. At the left boundary as the head of the expansion wave reflects, pressure and density continues to drop while velocity is zero at the boundary. The density

distribution shows that the contact surface and shock are about to cross. The program terminated and issued a message before the next output call was made, when the code determined that the contact surface and shock would cross. The exact density distribution compared with the computed density distribution in Fig. 4.4 clearly shows the shock and contact surface are modeled very accurately with a medium mesh. The expansion process, consisting of infinitesimal changes propagating along the characteristics is fairly well modeled. Figure 4.5 shows that tracking of the contact surface and shock wave, in comparison with exact locations at set times, is excellent. Using q+A and q-A based on conditions behind the burst diaphragm as estimates of the velocities of the head and tail of the expansion wave respectively results in positions of the head and tail slightly different from those of an exact Riemann solver code [Ref. 8]. The exact code is based on a method given in [Ref. 9:pp. 181-191]. However, an examination of the tabulated output showed that the solution method predicted changes in the gradients to occur at similar locations to those computed using the q+A and q-A estimates. Consequently, while there may be some inaccuracy in the numerical solution, the method of tracking the head and tail of the expansion wave is acceptable.

The ability to reproduce the same conditions but with flow to the left is demonstrated clearly in Fig. 4.6. The

entropy, density, velocity, and pressure distributions are mirror-images of those for the case of high pressure on the left.

Test Case 2 results in Fig. 4.7 show that at time zero with a pressure ratio at the boundary of 4, which is lower than the preset value inside the tube of 5, an expansion wave traveling inwards is generated. Outflow is seen in the negative velocity distribution developing near the left boundary, where negative velocity implies flow traveling to the left. The contact surface and shock are clearly formed to the right of the off-center diaphragm, and travel to the right. The expansion wave generated at the diaphragm travels to the left and intersects with the expansion wave generated at the left boundary. The pressure distribution shows the propagation of these waves as the pressure drops behind the head of each wave. With conditions reversed, so that the high pressure is to the right of the diaphragm, the results in Fig. 4.8 are a mirror-image with velocity negative for flow to the left, and positive for the outflow to the right.

Test Case 3 results in Fig. 4.9 demonstrate that the shock wave meeting an open boundary where the pressure is held constant, is correctly computed to exit the tube. The density distribution shows the jump increase across the shock, then another jump increase across the contact surface. Then density plunges down as an expansion wave

travels back into the tube when the shock exits. The pressure ratio at the open right boundary was held at unity, as seen in the pressure distribution which, at the boundary behaves similarly to the density distribution. The pressure at the boundary would be required to increase as the shock passed by, however the enforcement of the constant pressure locally causes an expansion wave to form traveling to the left. The entropy remains constant, at the value behind the shock, from the boundary back to the contact surface. There is a jump in entropy across the contact surface. The velocity distribution shows that as the shock travels to the right, the velocity jumps up. When the expansion wave generated at the right boundary travels inward the velocity continues to increase in magnitude while flowing to the right. The expansion wave generated at the diaphragm can be seen traveling to the left in the plots of pressure, density, and velocity. Reversing the situation and computing conditions for flow to the left, in Fig. 4.10, results in a mirror-image in the pressure, density, and entropy distributions. The velocity becomes negative since flow is to the left.

Test Case 4 is a repeat of the first test case but with a pressure ratio of 3.2 and a fine computational mesh. In Fig. 4.11 the density steps up across the shock and contact surface as they travel to the right. When the shock reflects the density jumps again. The velocity distribution

shows that the velocity drops to zero behind the reflected shock, after it had jumped up across the shock when it was traveling to the right. The instability seen in the velocity and pressure distributions near the midpoint occurred during the first 100 time steps. A numerical instability appears to generate at the diaphragm at time zero for pressure ratios less than about 3.0 which can be severe. For the conditions in this test case the transient instability damped out after the first 100 time steps. The shock, contact surface, and expansion wave are nevertheless seen to be accurately modeled. Figure 4.12 demonstrates that reversed conditions result in mirror-images of the computed conditions.

B. CURRENT LIMITATIONS OF THE CODE E1DV2

The current limitations of the E1DV2 code for modeling quasi-one-dimensional inviscid flow are in two categories. First, there are numerical limitations in obtaining solutions with the present code for different initial and boundary conditions. Second, the present coding limits what flow situations can be treated.

The numerical instability which occurs at low pressure ratios during the first few time steps can be severe. The assumption in the current code is that the shock forms immediately, which at high pressure ratios does not pose any problems. The mathematical modeling of the bursting diaphragm in subroutine "DBURST" adequately reduces the

transient instability for high pressure ratios, but not for low pressure ratios.

A second rather different numerical limitation was identified in numerically detecting the location of the head and tail of the expansion process. Because the expansion process is not a sharply defined front, fixing the precise locations of the head and tail waves numerically at a given time is difficult. However, the method currently used does accurately track the computed propagation of the wave along the characteristics. The characteristics are modeled currently as straight lines. A higher order curve would describe them more accurately.

The current code is limited to tracking two discontinuities and the expansion wave (i.e., a shock and a contact surface). Thus any situation which would generate another discontinuity can not be computed. The code can determine when this will occur, and will issue a message explaining the situation before terminating. Thus the shock colliding with a contact surface will currently terminate the program. A boundary pressure that is higher than the pressure inside the tube would also create a compression wave or possibly a shock wave. This condition will also terminate the program. Simulation of the process of a shock forming over an extended distance and time as compression waves pile on top of each other and strengthen, would also require additional code.

Finally, the variation of gamma that occurs across the contact surface in a wave rotor cannot be handled currently since E1DV2 assumes a single constant gamma throughout.

VI. CONCLUSIONS

Towards the development of a one-dimensional code for wave rotor applications, methods for tracking and correcting conditions across a contact discontinuity, applying open and closed-end boundary conditions, accounting for shock wave and contact surface interaction were devised and were presented here in detail. The EULER1 Fortran Code [Ref. 2] was revised to become the E1DV2 code with the following additional capabilities:

1) tracking of the contact surface and expansion wave
2) imposing high pressure initially on the right side of the diaphragm
3) correct jump conditions across the contact surface
4) allow open boundary conditions with constant pressure specified and allow exiting of shock and expansion waves
5) allow closed boundary conditions and model shock and expansion wave reflections
6) improved numerical accuracy from time zero to the maximum time step.

The code was tested on the shock tube problem under four different initial and boundary conditions with excellent results.

The following extensions are recommended to make the code suitable for wave rotor applications:

1) Solve the Riemann problem at the moment when the shock and contact surface intersect

2) Add additional code to track more than two discontinuities and one expansion wave

3) Incorporate a variable value of gamma into the code

4) Update the characteristic curve approximation from linear to a higher order polynomial

5) Add code necessary to handle open boundary conditions with inflow

6) Improve on the numerical computation at time zero for low pressure ratios across the diaphragm

7) Enable boundary conditions to be variable during program execution to allow wave rotor cycle design

8) Compare computations using the code with experimental data where available

9) Extend the code to two dimensions.

These extensions may require various degrees of effort. However, the ability of the QAZ1D solution method to be extended rather simply to describe viscous multi-dimensional flow suggests that such efforts would be justified.

APPENDIX A

DERIVATIONS OF EQUATIONS

A. LIST OF VARIABLES

A	Speed of sound
C_p	Specific heat at constant pressure
C_v	Specific heat at constant volume
e	specific internal energy
h	Specific enthalpy
P	Static pressure
Q	Modified Riemann variable
q_R	Reversible heat transferred
q	Velocity magnitude
R	Modified Riemann variable
R_G	Gas constant
S	Modified entropy (non-dimensional where $S = S \cdot R_G$)
T	Static temperature
t	Time
u	Velocity relative to a standing shock wave
v	Specific volume
W	Incoming Mach Number relative to a stationary shock wave
w	Work
ρ	Density
γ	Ratio of specific heats

B. DERIVATION OF SHOCK JUMP EQUATION FOR HIGH PRESSURE ON THE RIGHT

The analytical equation relating the non-dimensionalized extended Riemann variable, R, change through a normal shock is derived below similar to that for the Q variable change when high pressure is on the left [Ref. 2 :Appendix A].

Figure A.1 shows a shock moving to the left with velocity V_s. Subscript A is always associated with parameters to the left of any discontinuity, and B with those on the right. To enable normal shock relations to be used the system requires a velocity in the opposite direction but of equal magnitude be imposed upon it. The coordinate system was defined such that vectors, such as velocity, directed to the right are positive while those to the left are negative.

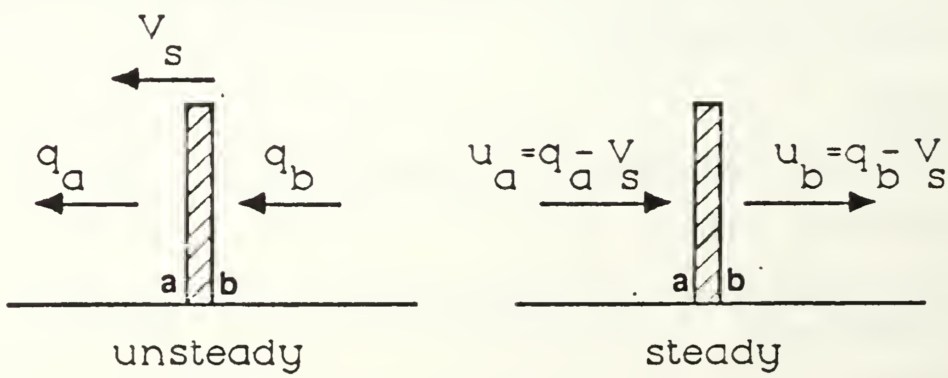


Figure A.1 Shock Wave with High Pressure on Right

Since relative incoming Mach Number, w, relates the velocity downstream, or that region into which the shock is traveling, to the speed of sound in that region [Ref. 10:pp. 114-154] then

$$w = \frac{q_A - V_s}{A_A} = \frac{u_A}{A_A} \qquad (A1)$$

w is a positive value because q_A, though negative, is smaller in magnitude than the equally negative shock velocity, V_s. The speed of sound, A, is positive by definition.

The appropriate extended Riemann variable in this case is R, defined as

$$R = q - AS \qquad (A2)$$

Since q is negative, this can be rewritten as

$$R = -(|q| + AS) \qquad (A3)$$

to emphasize that R is the Riemann variable associated with the lesser change across the shock [Ref. 3:pp. 18-21]. We adopt the pattern of non-dimensionalizing velocity by the speed of sound, pressure and density by corresponding values downstream of the shock. Using the entropy S, defined by Verhoff [Ref. 1:p. 2] as

$$S = \frac{2}{\gamma-1} - \frac{1}{\gamma(\gamma-1)} \ln(\frac{P}{\rho^{\gamma}}) \tag{A4}$$

then

$$\frac{R_B-R_A}{A_A} = \frac{q_B - A_B S_B}{A_A} - \frac{q_A - A_A S_A}{A_A}$$

$$= \frac{q_B - q_A}{A_A} + S_A - \frac{A_B}{A_A} S_B$$

$$= \frac{q_B - q_A}{A_A} + [\frac{2}{\gamma-1} - \frac{1}{\gamma(\gamma-1)} \ln(\frac{P_A}{P_A}(\frac{\rho_A}{\rho_A})^{\gamma})$$

$$- \frac{A_B}{A_A}[\frac{2}{\gamma-1} - \frac{1}{\gamma(\gamma-1)} \ln(\frac{P_B}{P_A}(\frac{\rho_A}{\rho_B})^{\gamma})]$$

$$= \frac{q_B - q_A}{A_A} + [1 - \frac{A_B}{A_A}] [\frac{2}{\gamma-1}] + (\frac{A_B}{A_A}) \frac{1}{\gamma(\gamma-1)}[\ln(\frac{P_B}{P_A}(\frac{\rho_B}{\rho_A})^{\gamma})] \tag{A5}$$

The ratios of pressure, density, and sonic velocity across a normal shock from Zucker [Ref. 7:p. 151] and Shapiro [Ref. 10:p. 118] in terms of Mach Number are

$$\frac{P_B}{P_A} = \frac{2\gamma}{\gamma+1} w^2 - \frac{\gamma-1}{\gamma+1} \tag{A6}$$

$$\frac{\rho_B}{\rho_A} = \frac{(\gamma+1)w^2}{(\gamma-1)w^2 + 2} \tag{A7}$$

$$\frac{A_B}{A_A} = \frac{1}{(\gamma+1)w}[2(\gamma-1)[1 + \frac{\gamma-1}{2} w^2] [\frac{2\gamma}{\gamma-1} w^2 - 1]]^{1/2} \tag{A8}$$

The first term on the right side of equation (A5) can be expressed in terms of w by applying simple continuity in mass.

$$\rho_A u_A \mathrm{Area}_A = \rho_B u_B \mathrm{Area}_B \tag{A9}$$

Taking the areas as equal, this becomes

$$\frac{\rho_A}{\rho_B} = \frac{u_B}{u_A} = \frac{(\gamma-1)w^2+2}{(\gamma+1)w^2} \tag{A10}$$

By subtracting one from each side

$$\frac{u_B}{u_A} - 1 = \frac{\gamma-1 \;\; w^2+ 2}{(\gamma+1)w^2} - 1$$

$$\frac{u_B - u_A}{u_A} = \frac{(\gamma-1)w^2 + 2 - (\gamma+1)w^2}{(\gamma+1)w^2}$$

$$\frac{u_B - u_A}{u_A} = \frac{2(1 - w^2)}{(\gamma+1)w^2} \tag{A11}$$

Substitute equation (A1) for u_A to get

$$\frac{u_B - u_A}{wA_A} = \frac{2(1 - w^2)}{(\gamma+1)w^2}$$

$$\frac{u_B - u_A}{A_A} = \frac{2(1 - w^2)}{(\gamma+1)w} \tag{A12}$$

thus

$$\frac{q_B - V_s - (q_A - V_s)}{A_A} = \frac{2(1 - w^2)}{(\gamma+1)w}$$

$$\frac{q_B - q_A}{A_A} = \frac{2(1 - w^2)}{(\gamma+1)w} \tag{A13}$$

Combining equations (A6), (A7), (A8), and (A13) into equation (A5) gives

$$\frac{R_B-R_A}{A_A} = \frac{2(1-w^2)}{(\gamma+1)w} + \left(\frac{2}{\gamma-1}\right)\{1-\left(\frac{1}{(\gamma+1)w}(2(\gamma-1)[1+\frac{\gamma-1}{2}w^2][\frac{2\gamma}{\gamma-1}w^2-1])^{1/2}\right)\}$$

$$+ \{\frac{1}{(\gamma+1)w}(2(\gamma-1)[1+\frac{\gamma-1}{2}w^2][\frac{2\gamma}{\gamma-1}w^2-1])^{1/2}\}$$

$$\times \{\frac{1}{\gamma(\gamma-1)}[\ln((\frac{2\gamma}{\gamma+1}w^2-\frac{\gamma-1}{\gamma+1})(\frac{(\gamma-1)w^2+2}{(\gamma+1)w^2})^{\gamma})]\} \tag{A14}$$

C. DERIVATION OF CONTACT SURFACE JUMP EQUATION

The analytical expression for the ratio of sonic velocity across a constant surface is derived, as follows:

The first law of thermodynamics is stated as

$$\begin{bmatrix}\text{change of}\\ \text{internal energy}\end{bmatrix} = \begin{bmatrix}\text{incremental}\\ \text{amount of}\\ \text{heat added}\\ \text{to system}\end{bmatrix} + \begin{bmatrix}\text{work done}\\ \text{on the}\\ \text{system by}\\ \text{surrounding}\end{bmatrix}$$

or, in differential form

$$de = \partial q_R + \partial W \tag{A15}$$

Internal energy is defined as

$$e = h - pv \tag{A16}$$

then

$$de = dh - pdv - vdp \tag{A17}$$

The incremental work is given by

$$\partial W = pdv \tag{A18}$$

The heat addition is determined from the definition of modified entropy,

$$d\bar{S} = -\frac{1}{\gamma}\frac{dq_R}{T}$$

$$dq_R = -\gamma T d\bar{S} \tag{A19}$$

Substituting equations (A17), (A18), and (A19) into equation (A15) yields

$$dh - pdv - vdp = -\gamma T d\bar{S} - pdv$$

Canceling like terms, and rearranging gives

$$dp = \frac{\gamma}{v} T d\bar{S} + \left(\frac{1}{v}\right) dh \tag{A20}$$

Enthalpy, h, is defined as

$$h = C_p T \tag{A21}$$

For a perfect gas the sonic velocity is given by

$$A = (\gamma R_G T)^{1/2}$$

thus

$$dA = \frac{1}{2}(\gamma R_G T)^{1/2}(dT/T) \qquad \text{(A22)}$$

Substituting equation (A21) and then (A22) into equation (A20), equation (A20) becomes

$$dp = \frac{\gamma}{v}Td\overline{S} + (\frac{1}{v})d(C_pT)$$

$$= \frac{\gamma}{v}Td\overline{S} + (\frac{1}{v})C_p dT$$

$$= \frac{\gamma}{v}Td\overline{S} + (\frac{1}{v})C_pT(\frac{2dA}{A}) \qquad \text{(A23)}$$

For an ideal gas,

$$\gamma = C_p/C_v$$

and

$$C_p = C_v + R_G$$

Combining these two equations, and rearranging

$$c_p = R_G \left(\frac{\gamma}{\gamma-1}\right) \tag{A24}$$

Substituting equation (A24) into equation (A23), and observing that pressure remains constant through the contact surface

$$\frac{T}{v}\gamma d\bar{S} + \frac{T}{v}\left(R_G\left(\frac{2\gamma}{\gamma-1}\right)\right)\frac{dA}{A} = 0 \tag{A25}$$

Eliminating T/v, equation (A25) becomes

$$\left(\frac{2}{\gamma-1}\right) d(\bar{S}/R_G) = -\frac{dA}{A} \tag{A26}$$

Then $\bar{S}/R_G$ becomes a non-dimensionalized entropy. Using the notation of $S = \bar{S}/R_G$ now for the non-dimensionalized form, so that by integrating both sides

$$\frac{2}{\gamma-1}\int_A^B dS = \int_A^B -\frac{dA}{A}$$

$$\frac{2}{\gamma-1}[S_B - S_A] = \ln A_A/A_B$$

which can be written as

$$A_A/A_B = \exp^{\left(\frac{(\gamma-1)}{2}(S_B-S_A)\right)} \tag{A27}$$

Definitions used in this development, excluding modified entropy were from [Ref. 11:pp. 87-125].

D. DERIVATION OF EXTENDED RIEMANN VARIABLE CHANGE ACROSS A CONTACT SURFACE

An analytical expression for the change in the non-dimensionalized extended Riemann variable, Q, across a contact surface traveling right is derived here. Figure A.2 depicts a contact surface moving right with velocity q.

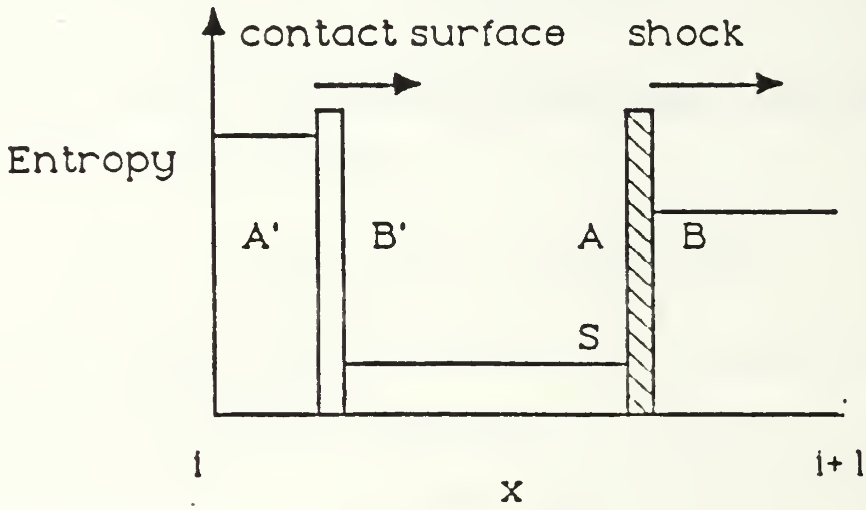


Figure A.2 Contact Surface Traveling Right Subscript Notation

Values of parameters to the left of the interface are denoted by subscript A', and those to the right with subscript B'. As illustrated, parameters to the left of the shock have subscript A, and to the right subscript B. All velocities are non-dimensionalized by sonic velocity immediately downstream of the discontinuity. Thus using the definition of the extended Riemann variable,

$$Q = q + AS \tag{A28}$$

then

$$\frac{Q_{A'} - Q_{B'}}{A_{B'}} = \frac{q_{A'} + A_{A'}S_{A'}}{A_{B'}} - \frac{q_{B'} + A_{B'}S_{B'}}{A_{B'}}$$

$$= \frac{q_{A'} - q_{B'}}{A_{B'}} + \frac{A_{A'}}{A_{B'}}(S_{A'}) - S_{B'} \tag{A29}$$

Since velocity is constant through a contact surface, so that

$$q_{A'} = q_{B'}$$

then equation (A29) becomes

$$\frac{Q_{A'} - Q_{B'}}{A_{B'}} = \frac{A_{A'}}{A_{B'}} S_{A'} - S_{B'}$$

Using equation (A27), this can be written

$$\frac{Q_{A'} - Q_{B'}}{A_{B'}} = \exp^{\left(\frac{(\gamma-1)}{2}(S_{B'} - S_{A'})\right)} S_{A'} - S_{B'} \tag{A30}$$

Multiply each side by $A_{B'}/A_{B}$, and since $A_{B'} = A_{A}$ then

$$\frac{Q_{A'} - Q_{B'}}{A_{B}} = [(\exp^{\left(\frac{(\gamma-1)}{2}(S_{B'} - S_{A'})\right)})(S_{A'}) - S_{B'}](A_{A}/A_{B}) \tag{A31}$$

For a contact surface traveling to the left, as illustrated in Fig. A.3, the derivation is entirely similar but

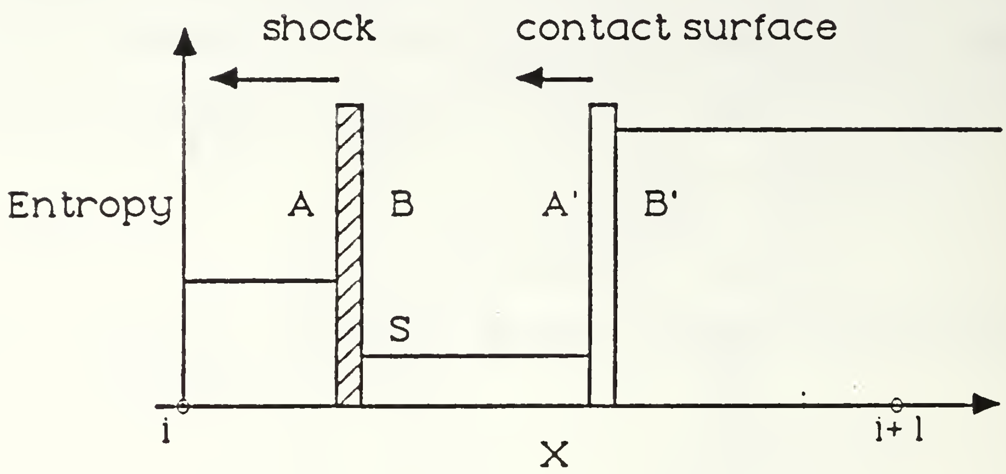


Figure A.3 Contact Surface Traveling Left Subscript Notation

using the R, extended Riemann variable with the result

$$\frac{R_{B'}-R_{A'}}{A_A} = [\exp^{(\frac{(\gamma-1)}{2})(S_{A'}-S_{B'})} (S_{B'})-S_{A'}] (\frac{A_B}{A_A}) \tag{A32}$$

E. DERIVATION OF OPEN BOUNDARY CONDITIONS WITH CONSTANT PRESSURE

The equation for density and temperature ratios in terms of pressure and entropy are derived below. The definition of modified entropy in non-dimensional form is

$$S = \frac{2}{\gamma-1} - \frac{1}{\gamma(\gamma-1)} \ln(\frac{P}{\rho^{\gamma}}) \tag{A33}$$

where P and ρ are non-dimensionalized ratios. Then

$$(S - \frac{2}{\gamma-1})(-\gamma(\gamma-1)) = \ln(P/\rho^{\gamma})$$

or

$$\exp^{[(S - \frac{2}{\gamma-1})(-\gamma(\gamma-1))]} = P/\rho^{\gamma}$$

Rearranging,

$$\rho^{\gamma} = [P[\exp^{[(S - \frac{2}{\gamma-1})(-\gamma(\gamma-1))]}]]$$

thus

$$\rho = P^{1/\gamma}(\exp^{(S - \frac{2}{\gamma-1})(-\gamma(\gamma-1))})^{-1/\gamma}$$

$$\rho = \{\frac{1}{P}(\exp^{(S - \frac{2}{\gamma-1})(-\gamma(\gamma-1))})\}^{-1/\gamma} \qquad (A34)$$

Using the perfect gas law,

$$\rho = \{\frac{1}{\rho T}(\exp^{(S - \frac{2}{\gamma-1})(-\gamma(\gamma-1))})\}^{-1/\gamma}$$

$$\rho^{-\gamma} = \frac{1}{\rho T}(\exp^{(S - \frac{2}{\gamma-1})(-\gamma(\gamma-1))})$$

multiply each side by T and divide by $\rho^{-\gamma}$ then

$$T = \rho^{\gamma}/\rho(\exp^{(S-\frac{2}{\gamma-1})(-\gamma(\gamma-1))})$$

thus

$$T = \rho^{\gamma-1}[\exp^{(S-\frac{2}{\gamma-1})(\gamma(1-\gamma))}] \tag{A35}$$

APPENDIX B

OPERATION OF E1DV2 ON THE NPS VM/CMS SYSTEM

A. TERMINAL REQUIREMENTS

Terminal requirements depend on the desired output. Any terminal, such as the IBM 3278, connected into the VM/CMS system is adequate for a tabular listing or Disspla Metafile of data. The Disspla Metafile stores graphical data for display using DISSPOP commands. If output of the plots on a monitor screen is desired, an IBM 3277-TEK618 dual screen terminal must be used. Table B.I lists terminal requirements for different outputs.

TABLE B.I

TERMINAL AND OUTPUT

OUTPUT	TERMINAL
Tabular listing of data	Any terminal
Graphical data in format for VERSATEC printing	Any terminal
Graphical data plotted on screen	IBM 3277-TEK618

B. PROCEDURE

1. Editing Variables and Program Setup

To change the initial and boundary conditions, graphical output, and grid size to run the program under

different conditions, the following must be done. First, pick the appropriate terminal from Table B.I and log on. Enter the editor by issuing the command

x E1DV2 FORTRAN A

Table B.II lists the code variables that will require editing and their meaning. In addition, to these variables if graphical plots are to be outputed then enter the "BORDER" and "EXACT" subroutines and edit the lines:

```
"FIRST ORDER   N = ???"
"DENSITY RATIO = ???  TEMP RATIO = ???"
"PRESSURE RATIO = ???"
```

To issue the output graphs to the screen comment out the lines "COMPRS", and use

CALL TEK618

Otherwise, if a Disspla Metafile of the graphics plot is desired, comment out the "TEK618" line, and use

CALL COMPRS

These lines are in the "Set up graphics plot of variable" loop in the main program.

To store these changes, hit the enter key, and type "file". Select the enter key once more. The program is now ready to be compiled and executed.

2. Commands

To compile and execute the E1DV2 code on the VM/CMS system perform the following commands exactly as typed:

TABLE B.II

EDITABLE VARIABLES

Variable	Edit
DIMENSION statement for arrays A(N),..., XARRAY(N)	set the dimension of the arrays equal to the number of nodes in the grid
N	set to the number of nodes in the grid
GRAPHS	set to: 0 for tabular listing of data 1 for plot of density, entropy, pressure and velocity distributions 2 for plot of exact density distribution compared to computed density distribution
SKIP	set to number of time steps between calls to output routines
JSTOP	set to maximum number of time steps
TRI	set to initial temperature ratio across diaphragm
PRI	set to initial pressure ratio across diaphragm
DRI	set to initial density ratio across diaphragm
QLI	set to initial velocity left of diaphragm
QRI	set to initial velocity right of diaphragm
LBDPRI	set to initial pressure ratio across left boundary
LBDTRI	set to initial temperature ratio across left boundary
LBDORI	set to initial density ratio across left boundary

RBDDRI	set to initial density ratio across right boundary
RBDPRI	set to initial pressure ratio across right boundary
RBDTRI	set to initial temperature ratio across right boundary
G	set to desired value of gamma
EE	set to desired error tolerance for calculation of characteristic slope (i.e., 0.1D-8)
LWPRES	set to: 2 for low pressure on right side of diaphragm 3 for low pressure on left side
LBNDRY RBNDRY	set to: 1 for closed boundary 0 for open boundary
LBDPRS RBDPRS	set to: 0 for constant pressure at left boundary 1 for pressure that adjusts at the left boundary
XINIT	set to location of diaphragm
VHEAD	set to exact velocity for head of expansion wave
VTAIL	set to exact velocity for tail of expansion wave
VCDE	set to exact velocity for contact surface
VSE	set to exact velocity for shock wave
DLCD	set to exact density behind contact surface
DLSH	set to exact density behind shock

SIGMA(1,2) SIGMA(2,2) SIGMA(3,2) SIGMA(4,2)	set to diaphragm location (i.e., 0.5D00)
Y	There are four cases where Y appears in the program edit as follows: Comment out Y = (Integer#), use: first, Y = (N+1)/2 if LWPRES = 2 second, Y = (N+3)/2 and third, Y = (N-1)/2 if LWPRES = 3 fourth, Y = (N+1)/2 for diaphragm at 0.5 Comment out those above, if diaphragm at another node. Set first, Y = (Integer #) of node for LWPRES = 2 second, Y = (# + 1) and third, Y = (Integer # - 1) of node for LWPRES = 3 fourth, Y = (#)

1) Increase the virtual memory by entering

DEFINE STORAGE 1M

2) To return to CMS environment enter

I CMS

3) To compile the program enter

FORTVS E1DV2

The screen will display messages as it compiles each routine and when finished a ready symbol appears.

4) To execute the program, enter

DISSPLA E1DV2

The message

...YOUR FORTRAN PROGRAM IS NOW BEING LOADED...
...EXECUTION WILL SOON FOLLOW...

should appear, followed by

...EXECUTION BEGINS...

If at a TEK618 terminal with GRAPHS equal to 1 or 2 then the screen on the TEK618 will begin plotting the selected graph. A

...press ENTER to continue...

message will appear on the 3277 terminal. If a copy of the plot is desired, do so now before pressing the enter key. After pressing the enter key on the 3277 terminal, the plot will be erased and the program will terminate. Proper termination will result in

END OF DISSPLAY 9.2 #### VECTORS IN 1 PLOT...

appearing, followed by a ready message.

If GRAPHS was set to 0, then proper termination would be a ready message. The tabular listing of pressure, density, velocity, entropy, and Riemann variables will be in "FILE FT09F001." The exact location of the shock, contact surface, and expansion wave with elapsed time will be in "FILE FT08F001." The computed location of the shock, contact surface, and expansion wave with elapsed time will be in "FILE FT10F001."

A second method to compile and execute the program, plus provide the files with a name is to create the following EXEC file on the user's disk.

```
FI  9 DISK FILE09 LISTING A(PERM
FI 10 DISK FILE10 LISTING A(PERM
FI  8 DISK FILE08 LISTING A(PERM
FORTVS &1
```

A possible file name and required file type would be

RUN EXEC

To compile the program and define the output files, enter

RUN E1DV2

After compiling is finished, and the ready message appears, enter

DISSPLA E1DV2

The program executes as outlined before.

APPENDIX C

FLOWCHARTS

Flowcharts for major routines in E1DV2 are shown.

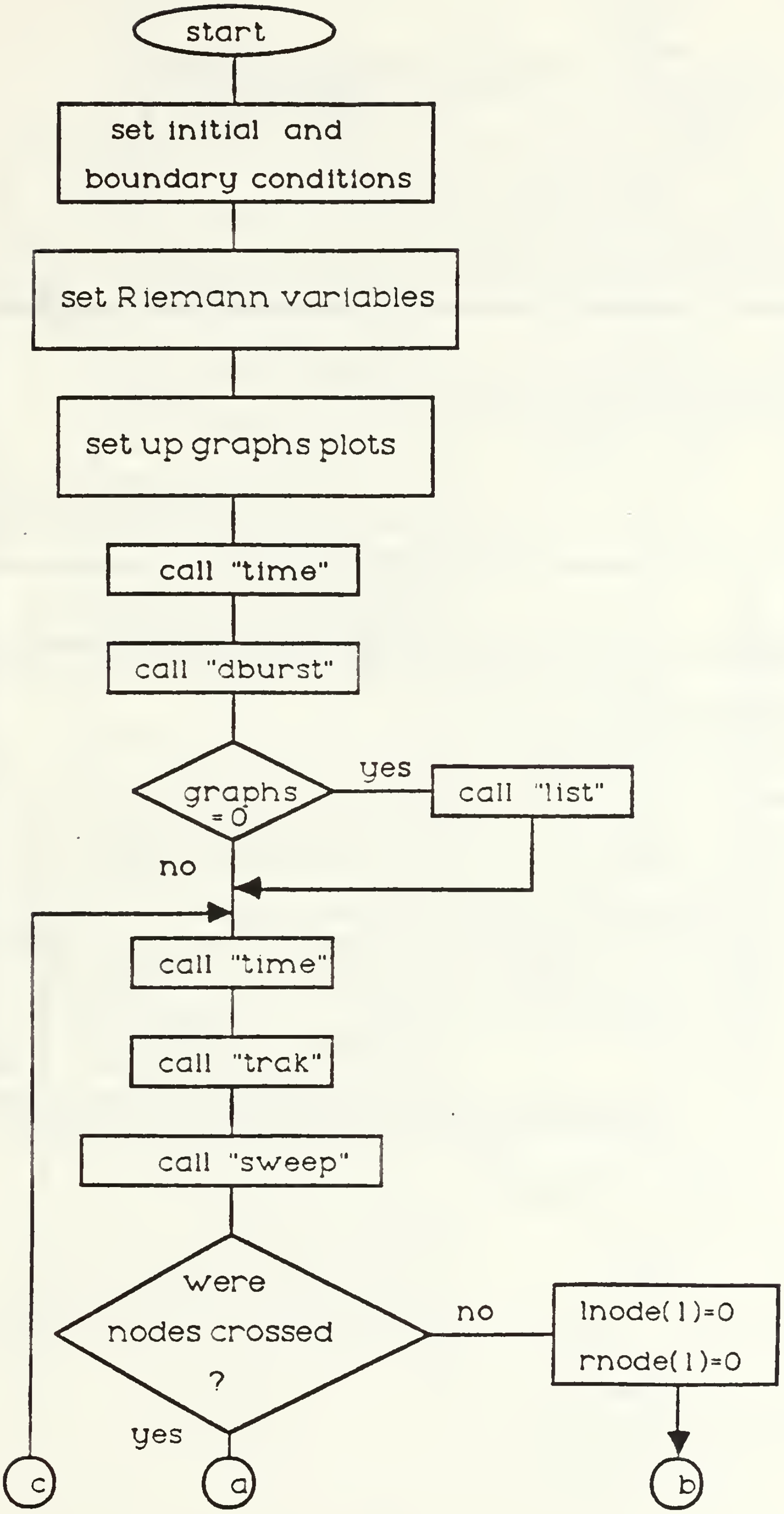


Figure C.1 Main Program Flowchart

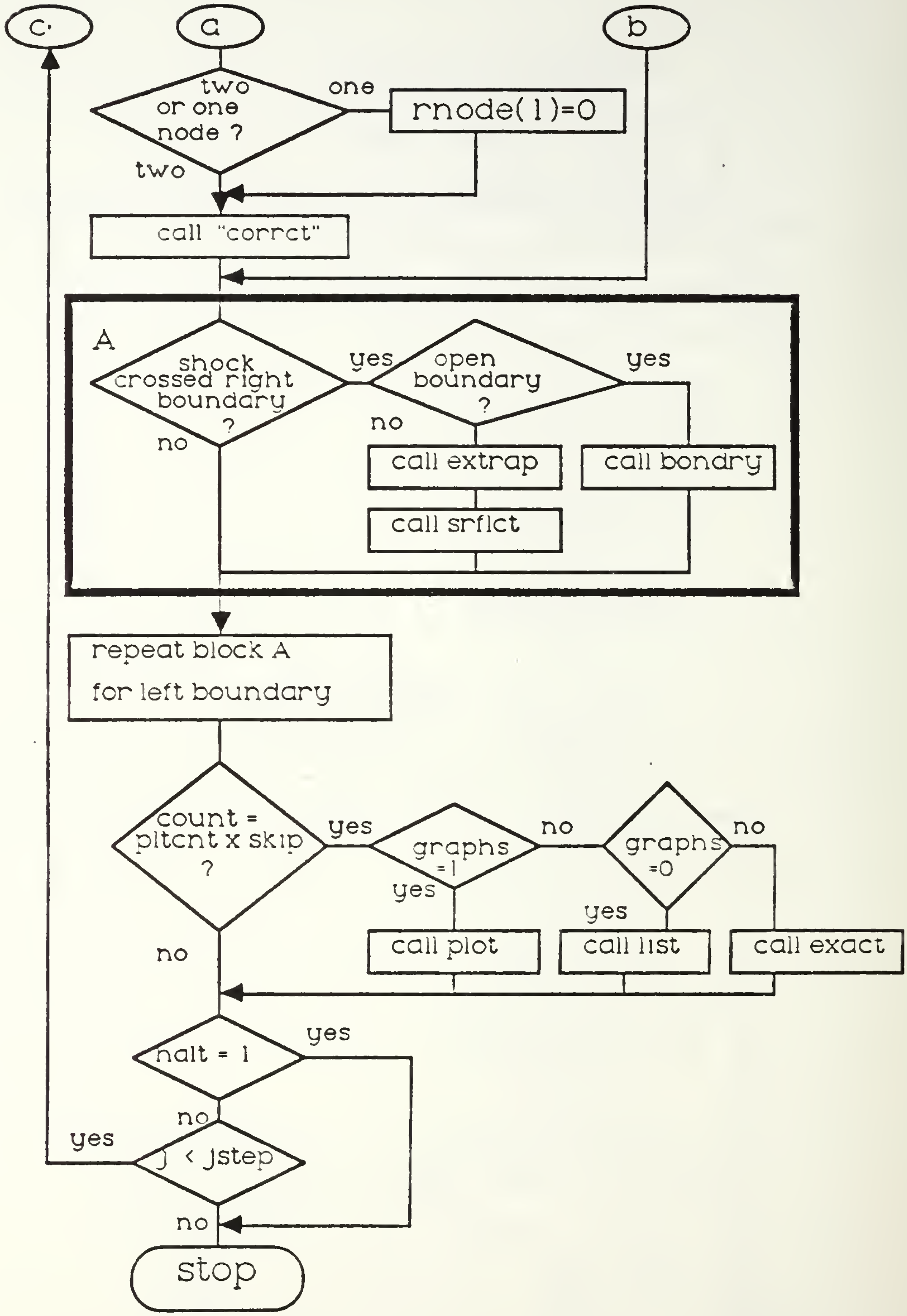


Figure C.1 (Continued)

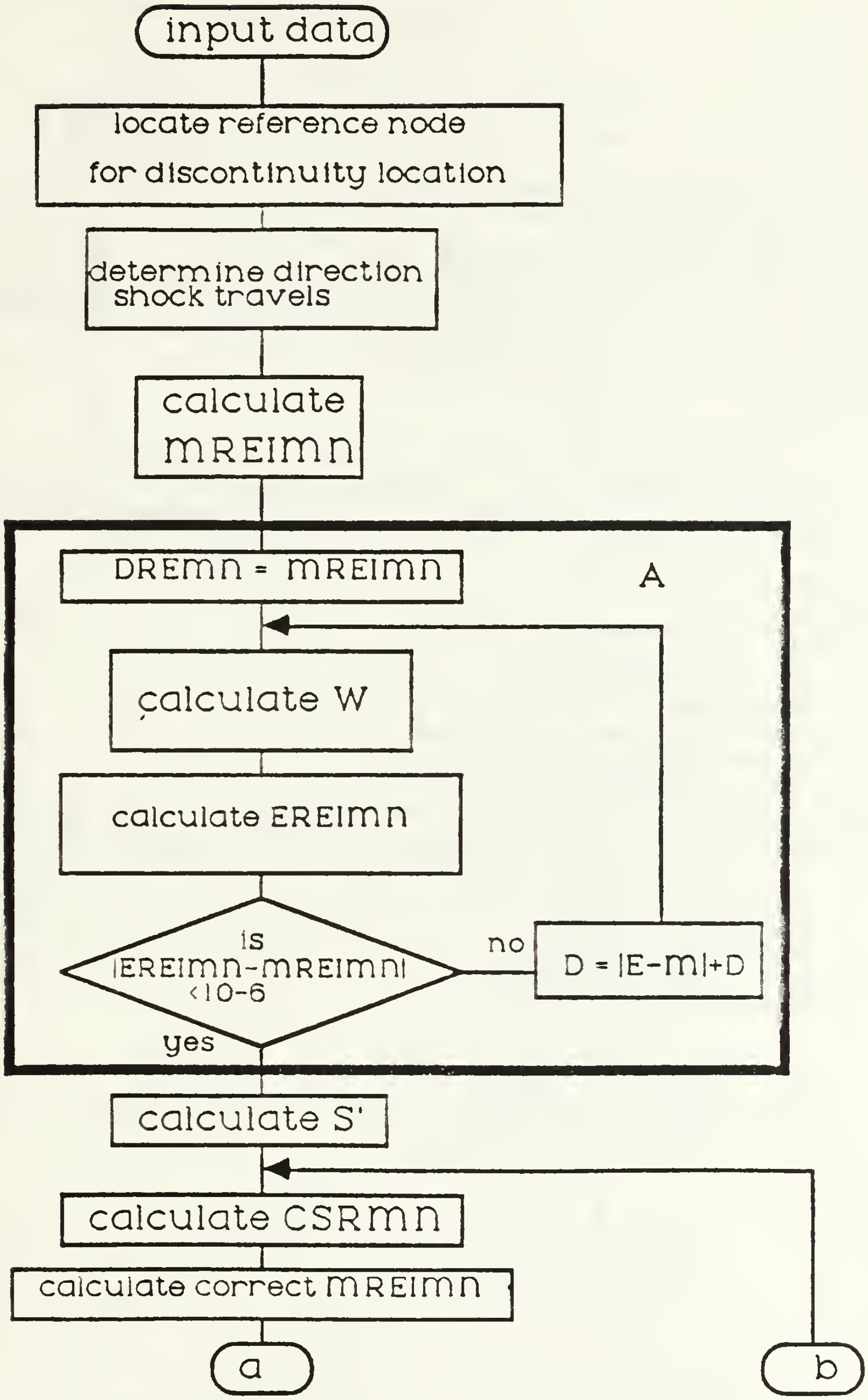


Figure C.2 "DBURST" Subroutine Flowchart

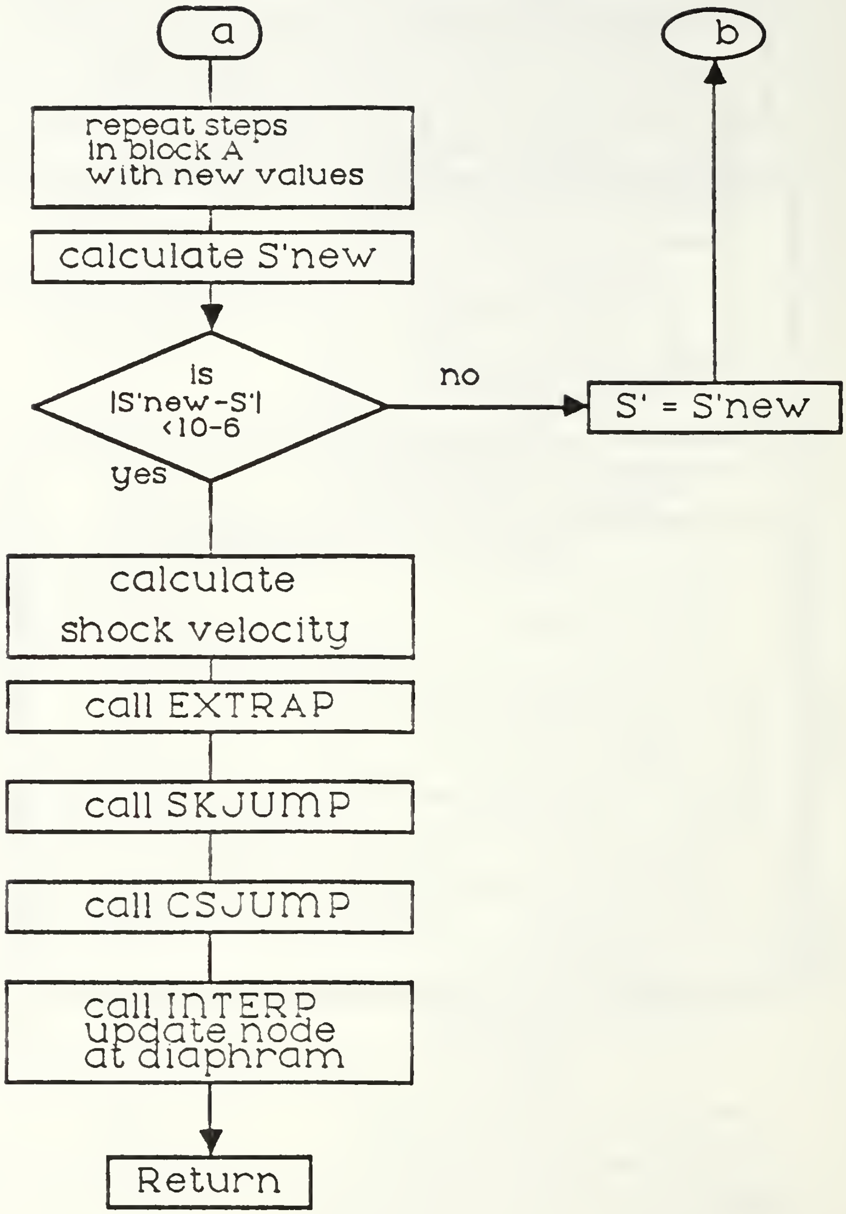


Figure C.2 (Continued)

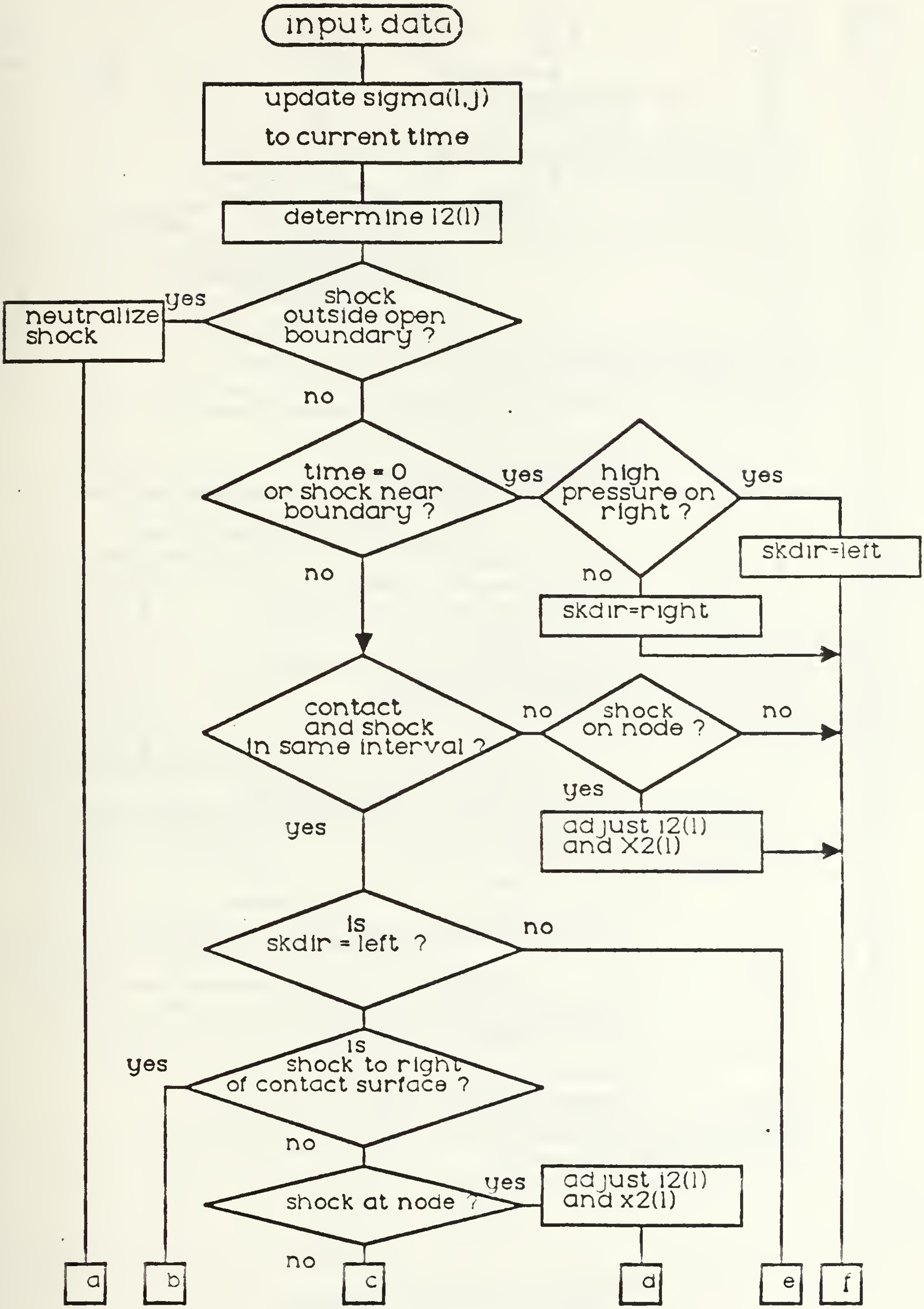


Figure C.3 "TRAK" Subroutine Flowchart

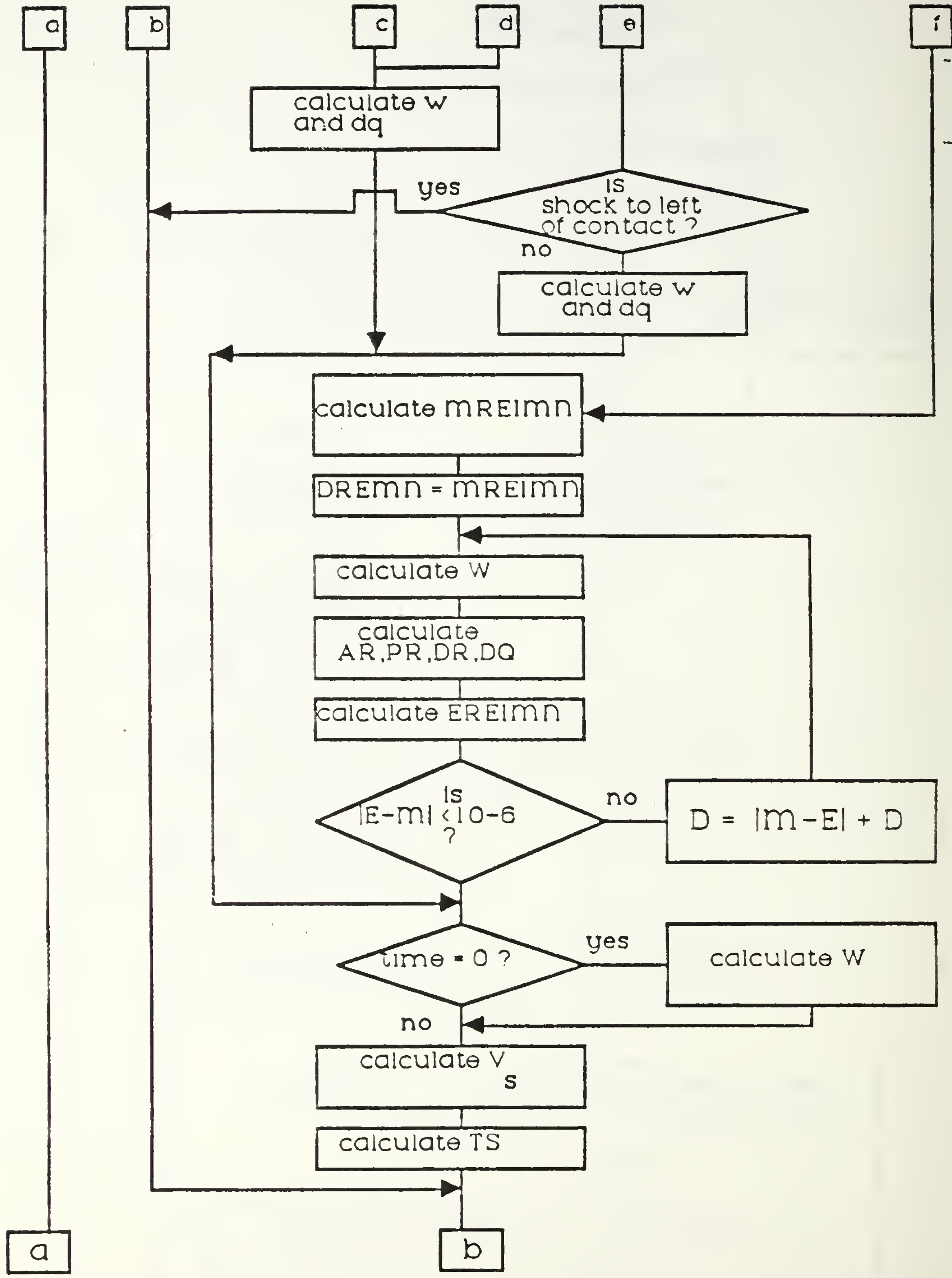


Figure C.3 (Continued)

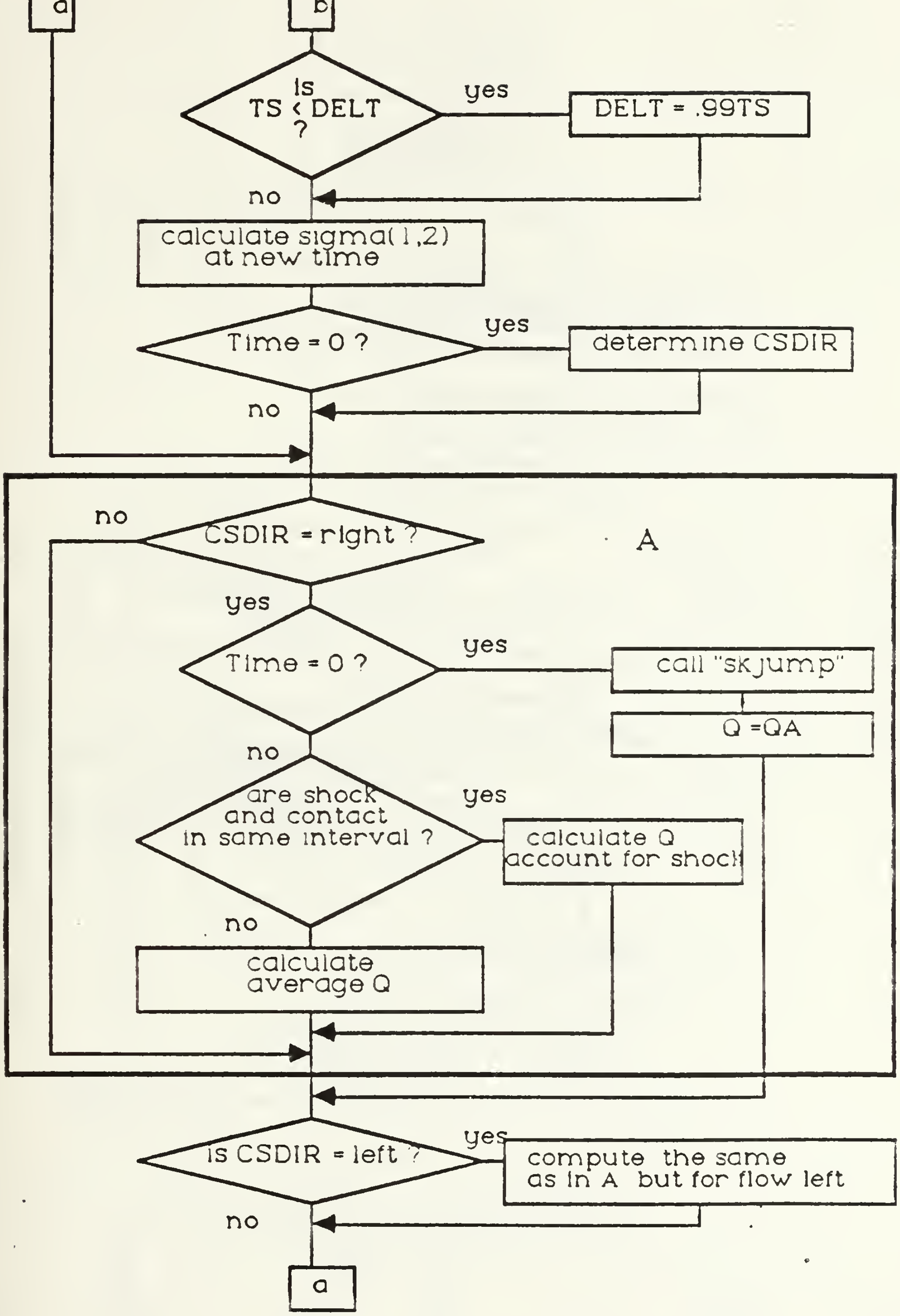


Figure C.3 (Continued)

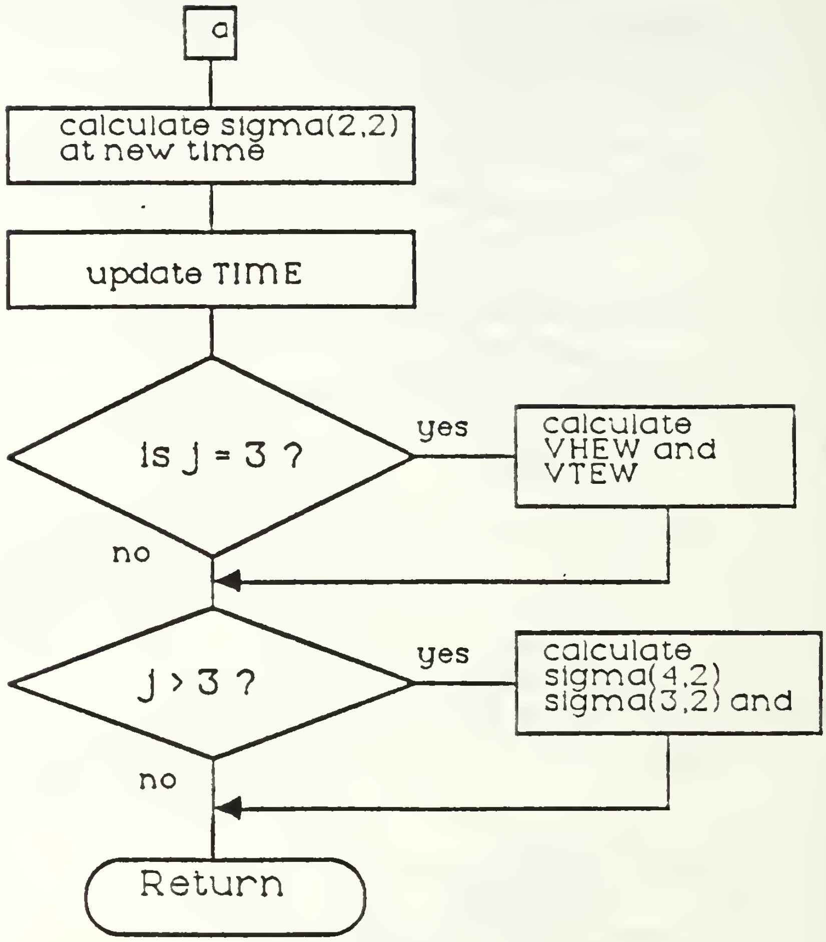


Figure C.3 (Continued)

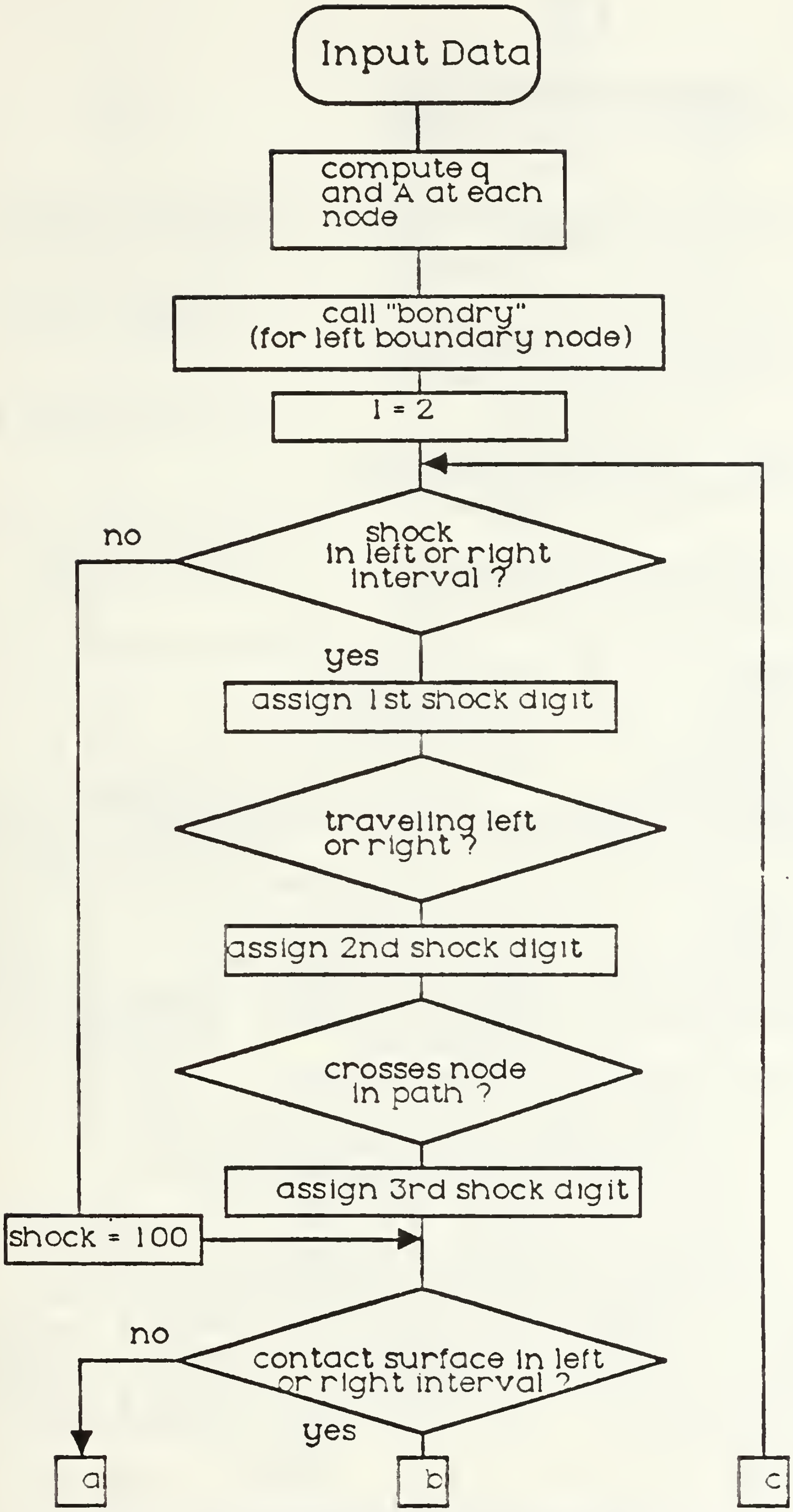


Figure C.4 "SWEEP" Subroutine Flowchart

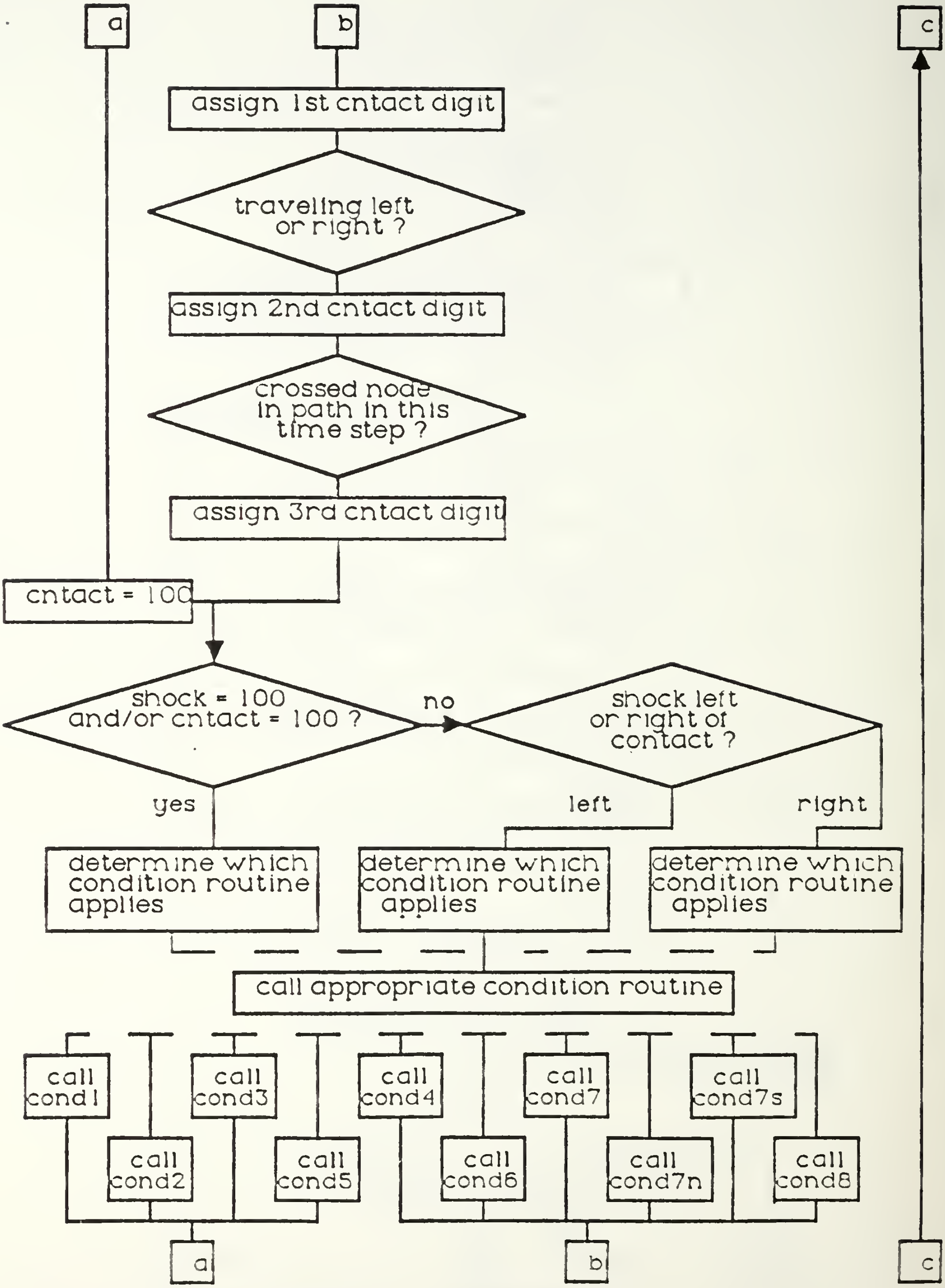


Figure C.4 (Continued)

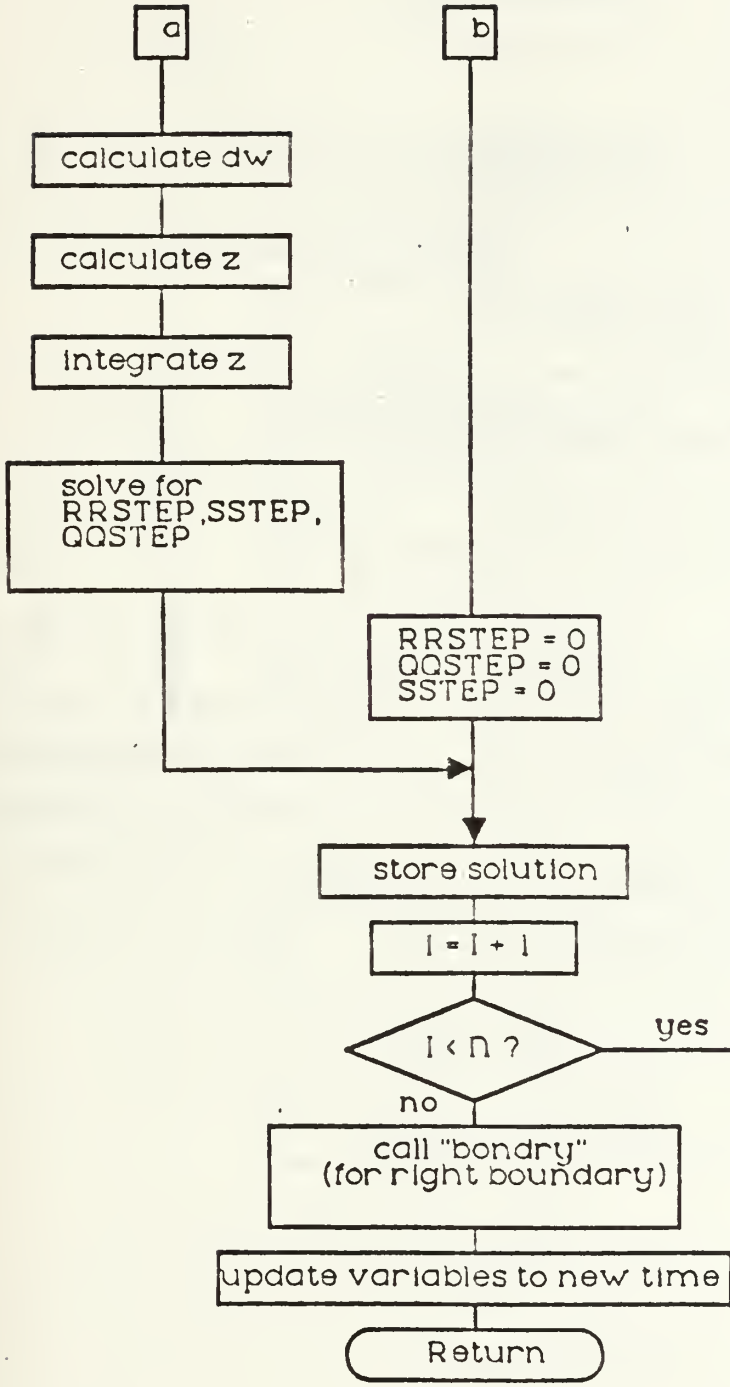


Figure C.4 (Continued)

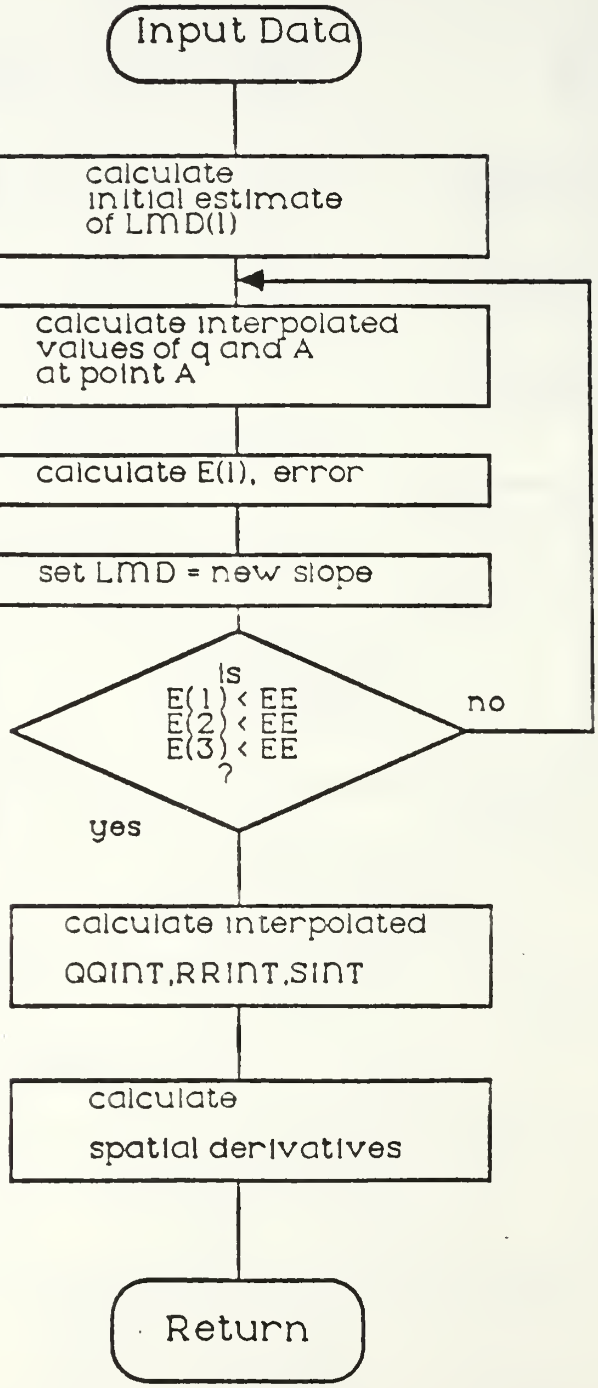


Figure C.5 General "COND1, 2, 3, and 5" Subroutine Flowchart

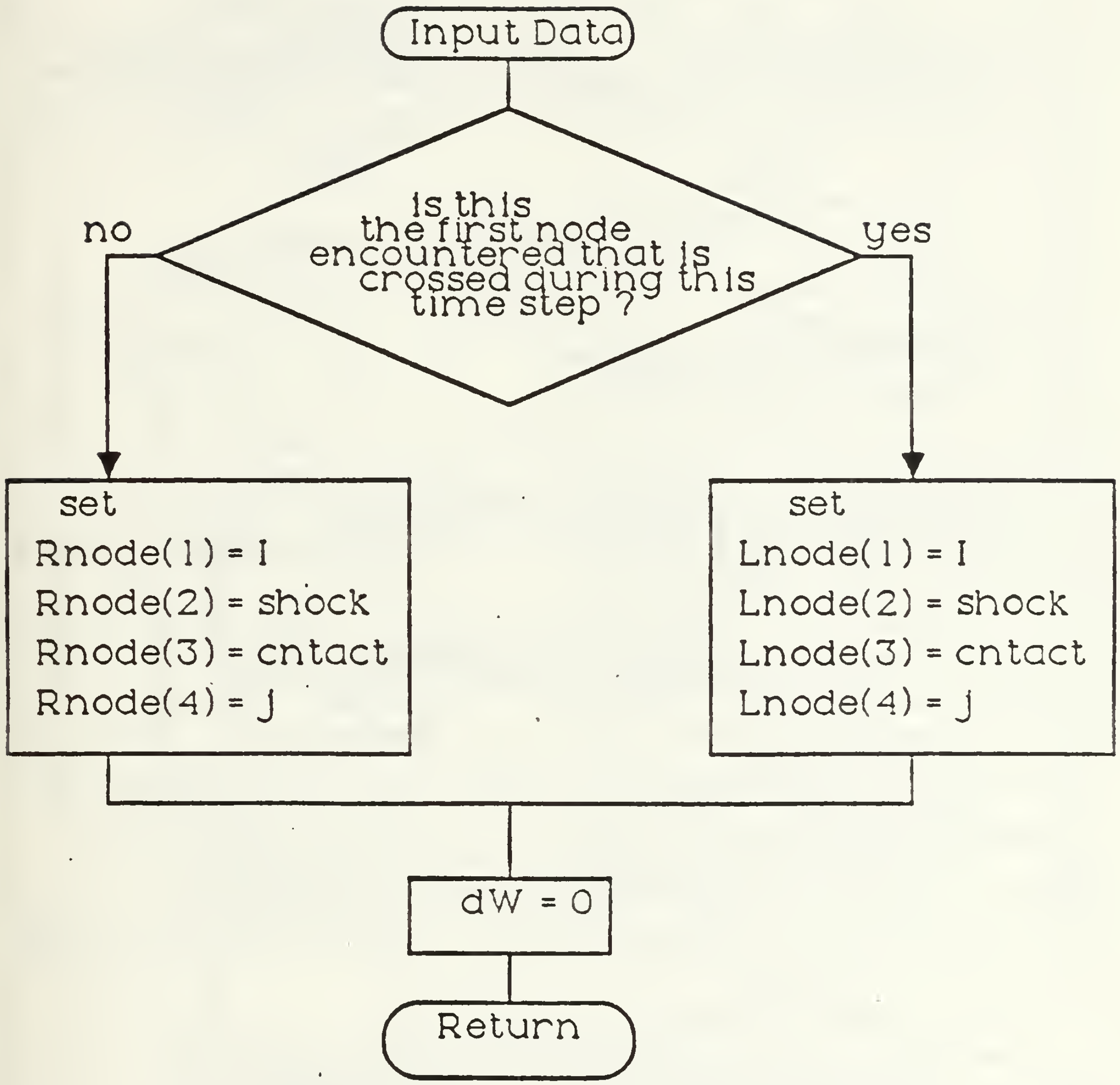


Figure C.6 "COND4" Subroutine Flowchart

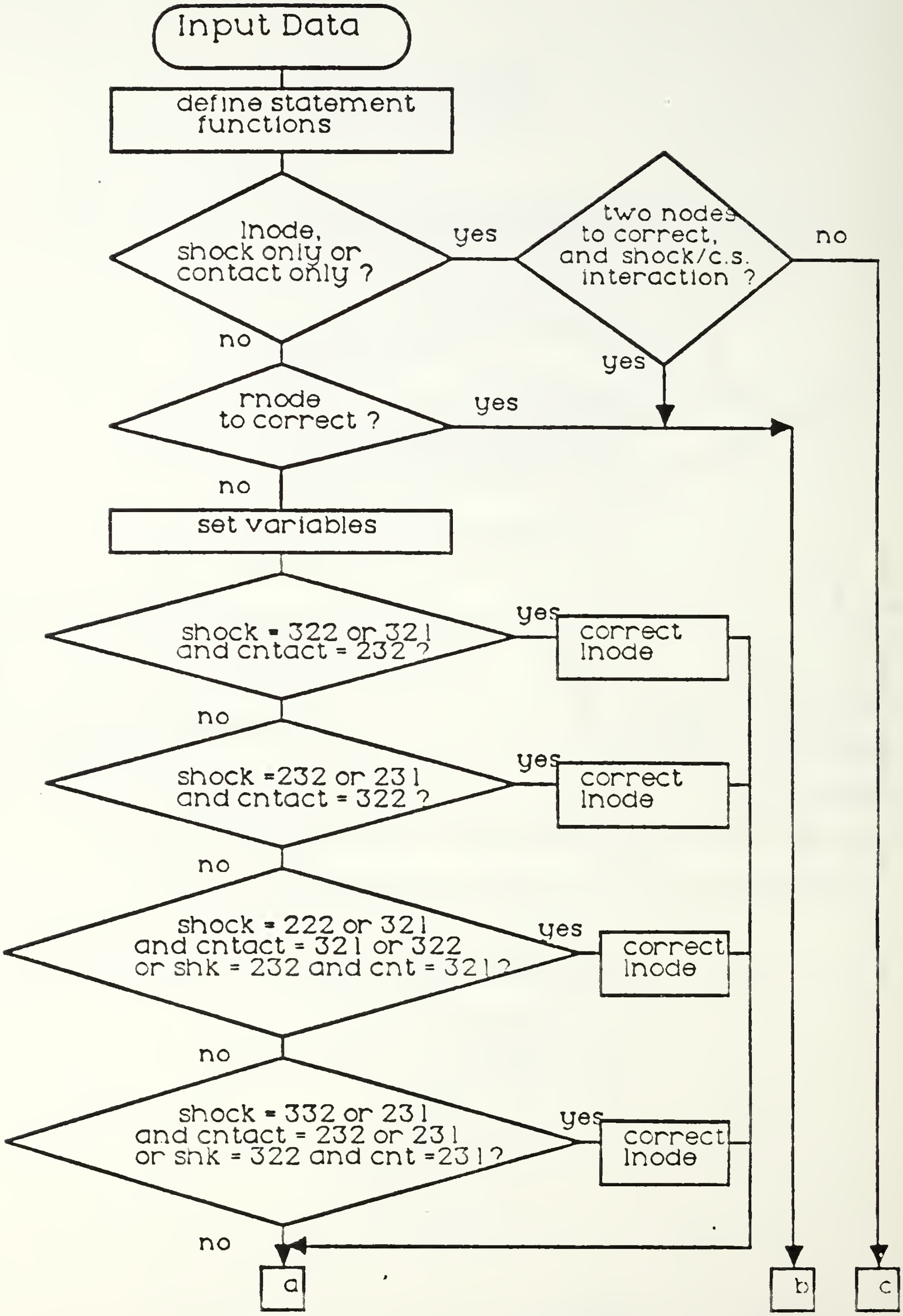


Figure C.7 "CORRCT" Subroutine Flowchart

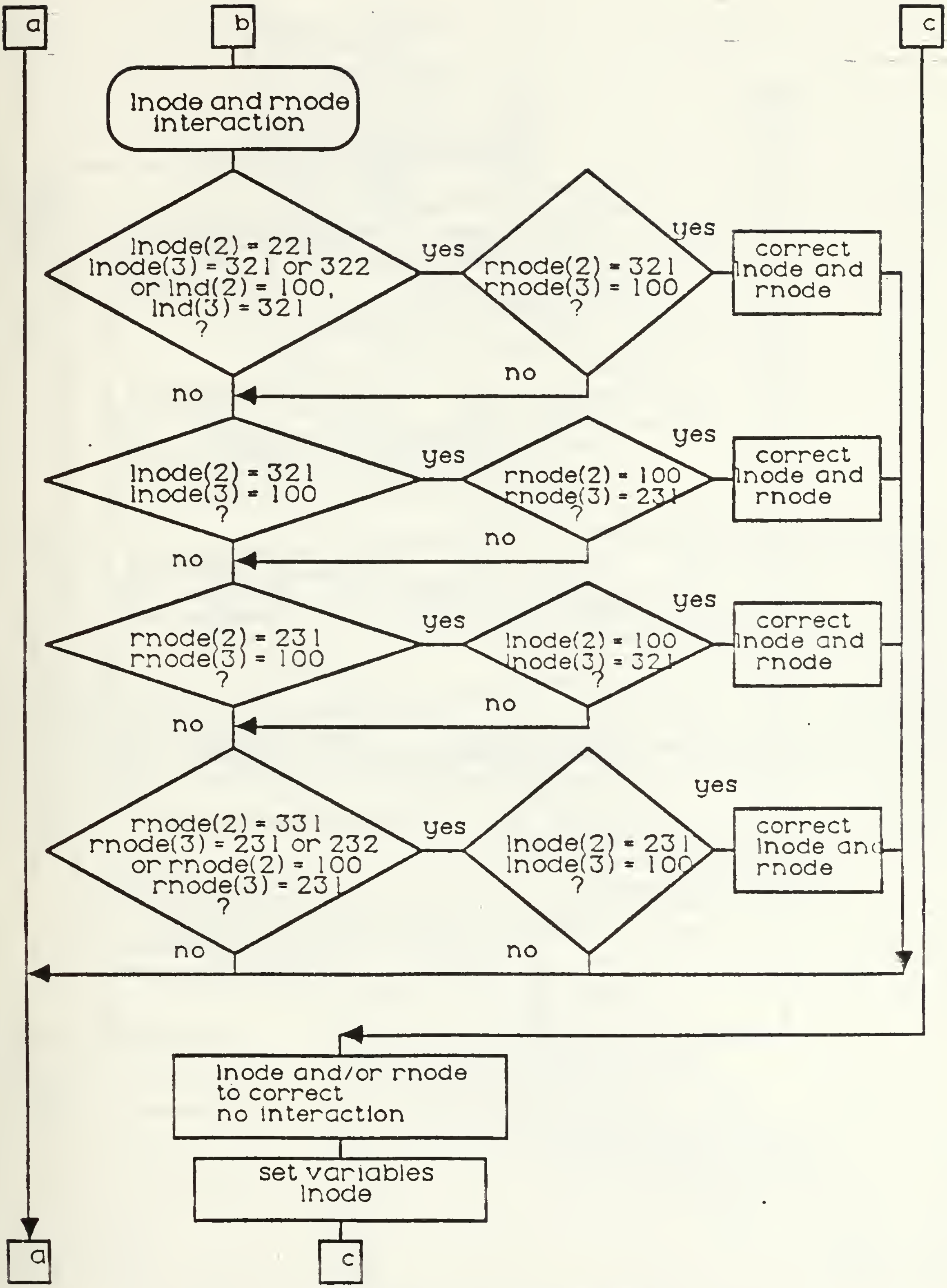


Figure C.7 (Continued)

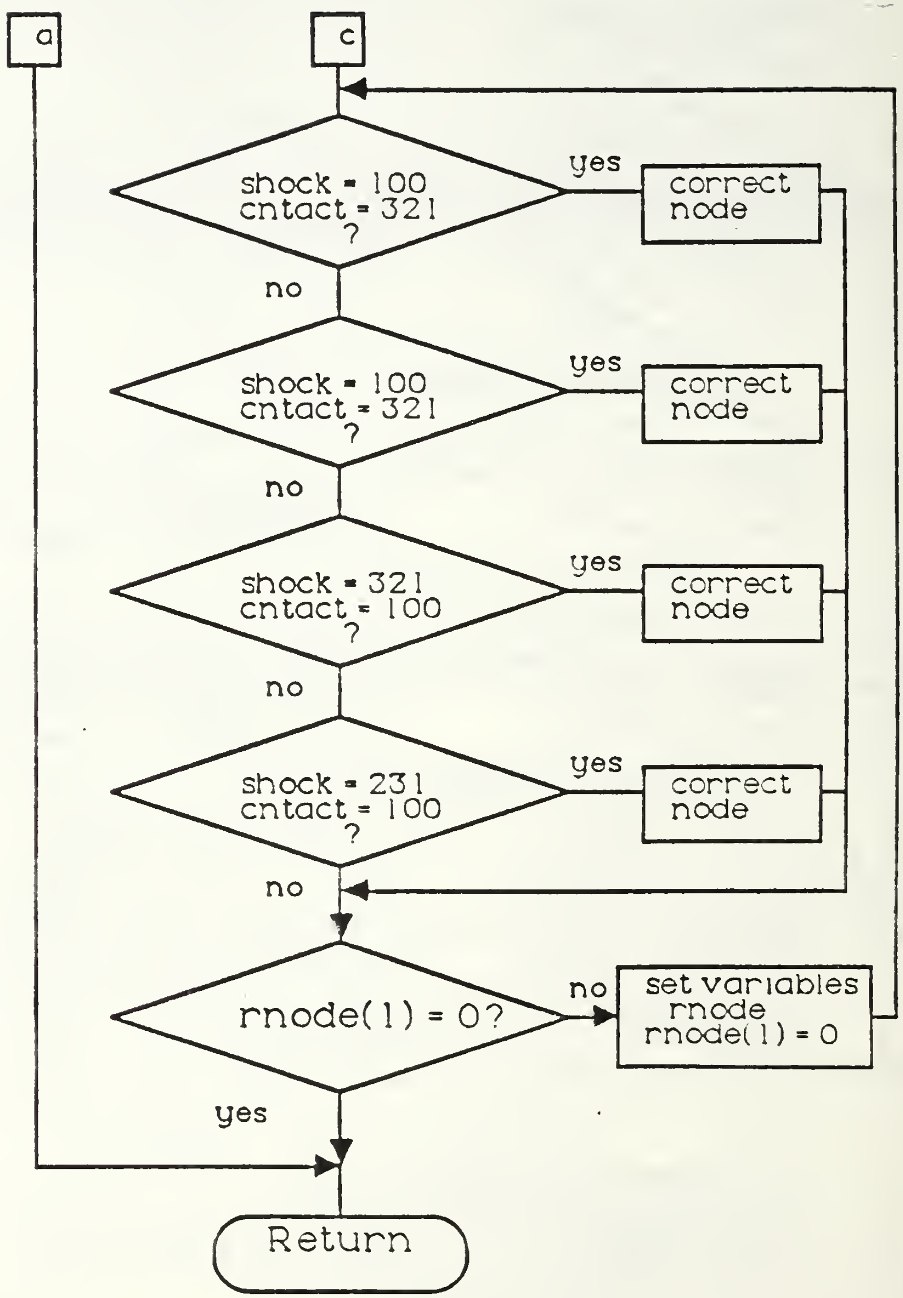


Figure C.7 (Continued)

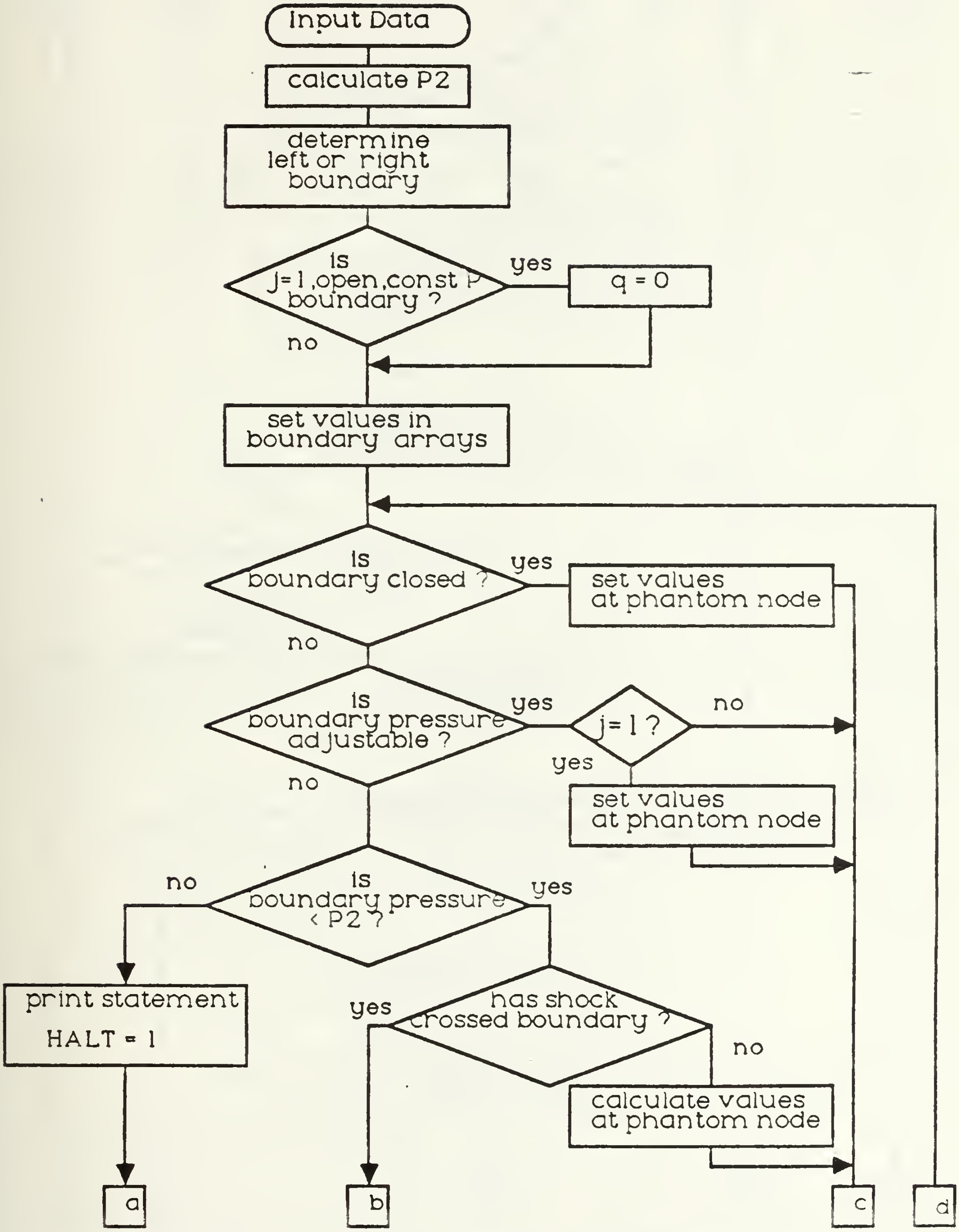


Figure C.8 "BONDRY" Subroutine Flowchart

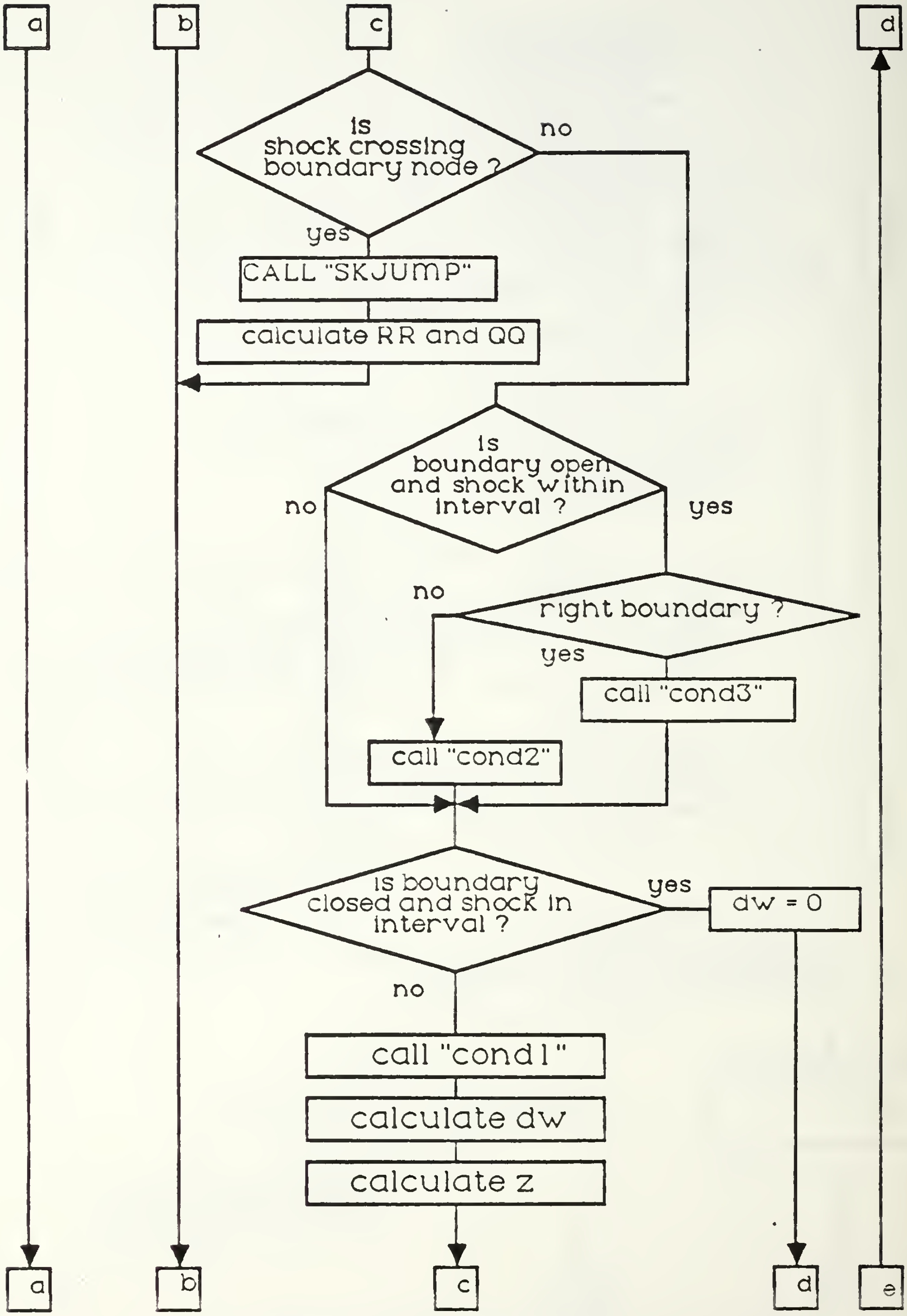


Figure C.8 (Continued)

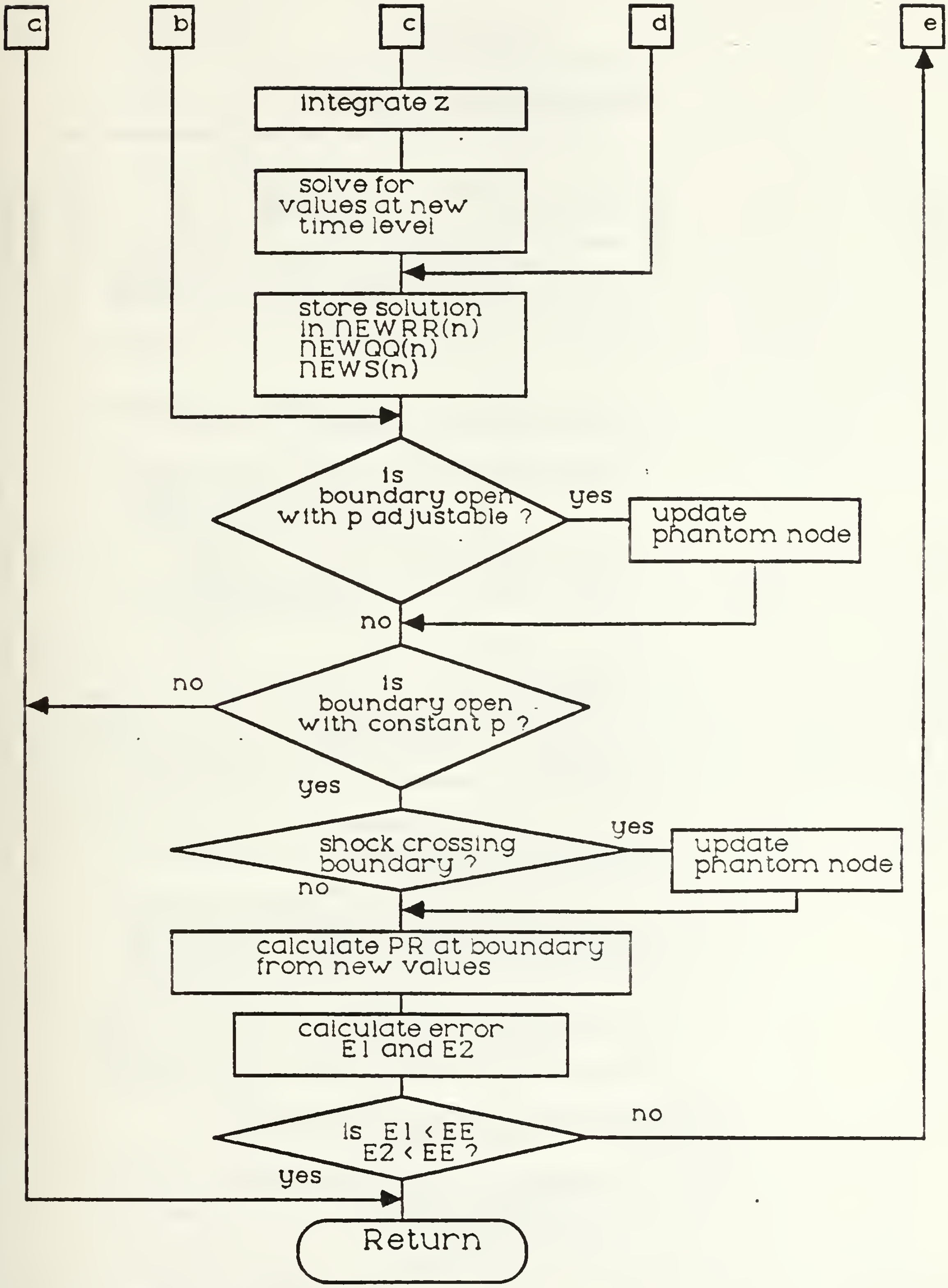


Figure C.8 (Continued)

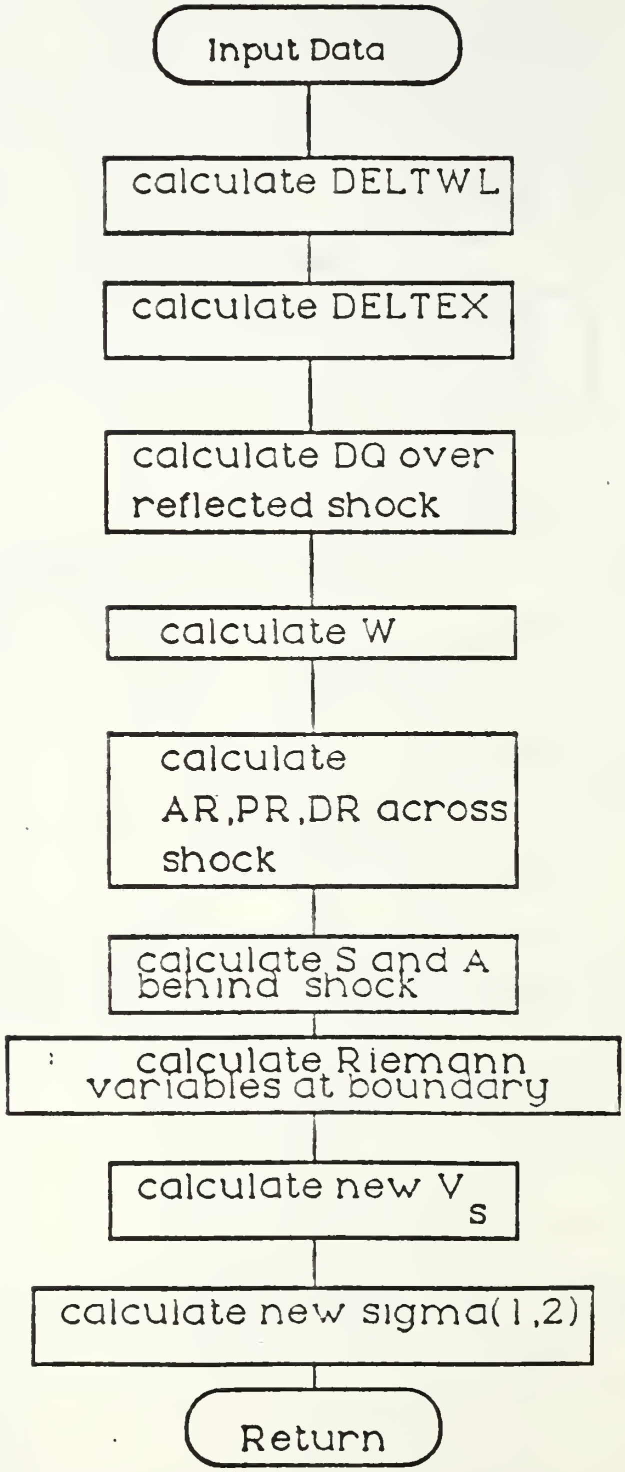


Figure C.9 "SRFLCT" Subroutine Flowchart

APPENDIX D

E1DV2 FORTRAN LISTING

```
C     ************************************************
C     *                                              *
C     *             E U L E R - 1 D                  *
C     *               VERSION 2                      *
C     *                (E1DV2)                       *
C     *                                              *
C     *    THIS PROGRAM SOLVES THE EULER EQUATIONS   *
C     *   EXPRESSED IN A QUAZI-ONE DIMENSIONAL       *
C     *   STREAMLINE COORDINATE SYSTEM.              *
C     *                                              *
C     *   AUTHOR - LT. D.T. JOHNSTON,FEB 1987        *
C     *                                              *
C     *                                              *
C     *   BASED ON THE EULER1 CODE BY                *
C     *   T.F. SALACKA, DEC 1985                     *
C     *                                              *
C     *   FEATURES OF THIS VERSION (2)               *
C     *   + ORDER OF SPATIAL DERIVITIVES - FIRST     *
C     *   + NUMBER OF SPATIAL DIMENSIONS - ONE       *
C     *   + DISCONTINUITIES TREATED:                 *
C     *          SHOCKS                  - YES       *
C     *          CONTACT DISCONTINUITIES - YES       *
C     *          EXPANSION WAVES         - YES       *
C     *   + HIGH PRESSURE SIDE
C     *          LEFT                    - YES
C     *          RIGHT                   - YES
C     *                                              *
C     ************************************************
C
C     ++++++++++++++++++++++++++++++++++++++++++++++++
C     +                                              +
C     +      CONVENTIONS   AND   DEFINITIONS         +
C     +                                              +
C     ++++++++++++++++++++++++++++++++++++++++++++++++
C
C     ------  NON-DIMENSIONING CONVENTION-----------
C
C       ALL VELOCITIES NON-DIM. BY THE SOUND SPEED ON
C           THE LOW PRESSURE SIDE OF THE DIAPHRAGM.
C       ALL PRESSURES, DENSITIES, AND TEMPERATURES ARE
C           NON-DIM. BY THEIR INITIAL VALUES ON THE LOW
C           PRESSURE SIDE OF THE DIAPHRAGM.
C       SPACIAL DISTANCE IS NON-DIM.BY OVERALL LENGTH.
C       ENTROPY IS NON-DIM. BY THE GAS CONSTANT, R.
C       TIME IS NON-DIM. BY (LENGTH/SOUND SPEED).
C       VELOCITIES AND DISTANCES ARE DEFINED POSITIVE
C           TO THE RIGHT.
C
C     ---------   SUBSCRIPT NOTATION----------------
C
C       I - SPACIAL NODE  (1 TO N FROM LEFT TO RIGHT)
C       J - TIME LEVEL   (0 IS THE INITIAL CONDITION)
C       K - DENOTES WHICH CHARACTERISTIC WAVE IS BEING
C           DEALT WITH:
C              1 = Q+A     2 = Q      3 = Q-A
C       L - DENOTES WHICH TYPE OF DISCONTINUITY IS
C           BEING DEALT WITH:
C              1 = SHOCK    2 = CONTACT DISCONTINUITY
C              3 = HEAD OF EXPANSION WAVE
C              4 = TAIL OF EXPANSION WAVE
C
```

```
C
C     A      - SPEED OF SOUND
C     *A    - DENOTES THE VALUE OF A VARIABLE AT THE NODE
C             TO THE LEFT OF A DISCONTINUITY.  * CAN
C             BE ANY VARIABLE NAME.
C     AR    - THE RATIO OF SOUND SPEED ACROSS A
C             SHOCK, A/B(SHOCK MOVING RIGHT),B/A(SHOCK MOVING LEFT)
C     *B    - DENOTES THE VALUE OF A VARIABLE AT THE NODE
C             TO THE RIGHT OF A DISCONTINUITY.
C     BDRY - 3 DENOTES LEFT BOUNDARY,2 THE RIGHT BOUNDARY
C     COUNT  - COUNTER FOR GRAPHICS ROUTINES
C     DARRAY - ARRAY OF DENSITY FOR PLOTTING
C     DELT   - TIME STEP
C     DLCD   - DENSITY BEHIND THE CONTACT
C               DISCONTINUITY IN THE EXACT SOLUTION.
C     DLSH   - DENSITY BEHIND THE SHOCK IN THE
C               EXACT SOLUTION.
C     DQ    - THE JUMP IN VELOCITY ACROSS THE SHOCK
C             DIVIDED BY THE SOUND SPEED AT B(RIGHT) OR A(LEFT)
C     DRI    - INITIAL DENSITY RATIO ACROSS THE SHOCK
C     EE     - DESIRED PRECISION FOR CHARACTERISTIC CALCULATIONS
C     G      - GAMMA (RATIO OF SPECIFIC HEATS)
C     GRAPHS - FOR GRAPHICAL OUTPUT,  0=NONE (TABULAR)
C               1=PLOTS ALL VARIABLES
C               2=COMPARES DENSITY WITH EXACT SOLUTION
C     G1     - 1/(G-1)
C     G2     - 2/(G-1)
C     H      - 1/(N-1)
C     HALT - TERMINATES PROGRAM IF 1,SET BY CONDITIONS NOT CODED
C     I2    - NUMBER OF THE NODE TO THE RIGHT OF A
C             DISCONTINUITY.
C     JSTOP  - NUMBER OF TIME LEVELS TO BE CALCULATED
C     LBDDR - LEFT BOUNDARY DENSITY RATIO
C     LBDDRI - LEFT BOUNDARY DENSITY RATIO AT TIME ZERO
C     LBDPR - LEFT BOUNDARY PRESSURE RATIO
C     LBDPRI - LEFT BOUNDARY PRESSURE RATIO AT TIME ZERO
C     LBDPRS - VALUE OF 0 DENOTES CONSTANT PRESSURE AT LEFT BOUNDARY
C              ,1 DENOTES ADJUSTABLE PRESSURE AT THE LEFT BOUNDARY
C     LBDTR - LEFT BOUNDARY TEMPERATURE RATIO
C     LBDTRI - LEFT BOUNDARY TEMPERATURE RATIO AT TIME ZERO
C     LBNDRY - DENOTES LEFT BOUNDARY CONDITION, OPEN OR CLOSED
C     LNODE - ARRAY OF LEFT MOST NODE TO BE CORRECTED IN CORRCT
C     LWPRES - DENOTES WHICH SIDE OF DIAPHRAGM HAS LOW PRESSURE
C     N      - NUMBER OF SPACIAL NODES (ODD NUMBER)
C     ND     - DOUBLE PRECISION VALUE OF N
C     NEW**(I)- STORED VALUES OF ** FOR THE NEXT TIME LEVEL
C     PARRAY - ARRAY OF PRESSURES FOR PLOTTING
C     PRI    - INITIAL PRESSURE RATIO ACROSS THE SHOCK
C     PLTCNT - COUNTER FOR GRAPHICS ROUTINES
C     Q      - ABSOLUTE FLUID VELOCITY
C     QARRAY - ARRAY OF VELOCITIES FOR PLOTTING
C     QLBD - INITIAL VELOCITY AT LEFT BOUNDARY
C     QLI    - INITIAL VELOCITY LEFT OF THE DIAPHRAGM
C     QRBD - INITIAL VELOCITY AT RIGHT BOUNDARY
C     QRI    - INITIAL VELOCITY RIGHT OF THE DIAPHRAGM
C     QQ     - Q+A*S (EXTENDED RIEMANN VARIABLE)
C     RBDDR - RIGHT BOUNDARY DENSITY RATIO
C     RBDDRI - INITIAL RIGHT BOUNDARY DENSITY RATIO
C     RBDPR - RIGHT BOUNDARY PRESSURE RATIO
C     RBDPRI - INITIAL RIGHT BOUNDARY PRESSURE RATIO
C     RBDPRS - VALUE OF 0 DENOTES A CONSTANT PRESSURE AT RIGHT
C              BOUNDARY, WHILE 1 IS FOR ADJUSTABLE PRESSURE
C     RBDTR - RIGHT BOUNDARY TEMPERATURE RATIO
C     RBDTRI - INITIAL RIGHT BOUNDARY TEMPERATURE RATIO
C     RBNDRY - DENOTES RIGHT BOUNDARY OPEN OR CLOSED
C     RNODE - ARRAY FOR RIGHT MOST NODE TO BE CORRECTED IN CORRCT
C     RR     - Q-A*S (EXTENDED RIEMANN VARIABLE)
C     S      - ENTROPY
C     SARRAY - ARRAY OF ENTROPY FOR PLOTTING
```

```
C     SIGMA  - SPATIAL LOCATION OF DISCONTINUITIES
C               SIGMA(L,J) WHERE L INCICATES THE TYPE OF
C                 DISCONTINUITY  AND J INDICATES THE
C                 TIME LEVEL;  1 - CURRENT LEVEL
C                 2 - LEVEL BEING CALCULATED
C     SK - INTEGER THAT DENOTES RELATIVE LOCATION OF SHOCK NEAR
C          BOUNDARIES
C     SKIP    - VARIABLE WHICH INDICATES HOW MANY TIME STEPS BETWEEN
C               CALLS TO OUTPUT ROUTINES
C     T       - TIME SINCE INITIAL CONDITIONS
C     TRI     - INITIAL TEMP. RATIO ACROSS THE SHOCK
C     VHEAD   - VELOCITY OF THE HEAD OF THE EXPANSION
C                WAVE FOR THE EXACT SOLUTION.
C     VTAIL   - VELOCITY OF THE TAIL OF THE EXPANSION
C                WAVE FOR THE EXACT SOLUTION.
C     VCDE    - VELOCTIY OF THE CONTACT DISCONTINUITY
C                FOR THE EXACT SOLUTION.
C     VS    - THE SHOCK SPEED(POSITIVE RIGHT, NEGATIVE LEFT)
C     VSE     - VELOCITY OF THE SHOCK FOR THE EXACT
C                SOLUTION.
C     W       - MACH NO. RELATIVE TO A STANDING SHOCK
C     XARRAY - ARRAY OF SPATIAL POSITIONS FOR PLOTTING
C     XEXACT - ARRAY OF SIX X VALUES FOR THE EXACT SOLUTION.
C     XINIT   - INITIAL POSITION OF DISCONTINUITY FOR
C                EXACT SOLUTION PLOTTING.
C     X2      - LOCATION OF NODE TO RIGHT OF DISCONT.
C               ALONG THE SPACIAL AXIS.
C     Y       - (N+1)/2
C     YEXACT - ARRAY OF SIX DENSITY VALUES FOR THE EXACT SOLUTION.
C
C     *** OTHER VARIABLES ARE DEFINED IN THE
C               SUBROUTINES WHERE THEY ARE USED ****
C
C     ++++++++++++++++++++++++++++++++++++++++++++++++
C     +                                              +
C     +              PROBLEM  SET - UP               +
C     +                                              +
C     ++++++++++++++++++++++++++++++++++++++++++++++++
C
C     THE PARTICULAR PROBLEM FOR THIS VERSION IS:
C
C         SHOCK TUBE, SINGLE CENTERED DIAPHRAGM WITH
C         HIGH PRESSURE SIDE TO THE RIGHT.
C
C         BOUNDARY CONDITIONS - LEFT END CLOSED,RIGHT END OPEN
C
C     ---------  VARIABLE  DECLARATIONS  -------------
C
      DIMENSION I2(4),X2(4),XEXACT(6),YEXACT(6),LNODE(4),RNODE(4),
     C          SIGMA(4,2)
C
C     +++++++ USER INPUT REQUIRED HERE +++++++++++++++
C
C ----- SET THE DIMENSIONS EQUAL TO N -----
C
      DIMENSION A(101),Q(101),QQ(101),RR(101),S(101),
     C          NEWRR(101),NEWS(101),NEWQQ(101),
     C          PARRAY(101),DARRAY(101),SARRAY(101),
     C          QARRAY(101),XARRAY(101)
C
      INTEGER I,J,N,JSTOP,Y,GRAPHS,COUNT,PLTCNT,BDRY,SK,RBNDRY,
     C        SKIP,I2,LWPRES,HALT,LNODE,RNODE,LBNDRY,LBDPRS,RBDPRS
      DOUBLE PRECISION TRI,PRI,QLI,QRI,DRI,G,G1,G2,SIGMA,EE,NEWQQ,
     C                DELT,H,ND,X2,AR,W,DQ,VS,T,A,Q,QQ,RR,S,NEWRR,NEWS,
     C                LBDPRI,LBDTRI,LBDDRI,LBDPR,LBDDR,LBDTR,QLBD,
     C                RRA,QQA,AA,SA,QA,RRB,QQB,AB,SB,QB,
     C                RBDPRI,RBDTRI,RBDDRI,RBDPR,RBDDR,RBDTR,QRBD,
     C                QCS,VTEW,VHEW
      REAL VTAIL,VCDE,VSE,DLSH,DLCD,XINIT,VHEAD,
```

```
     C     XEXACT,YEXACT,PARRAY,DARRAY,SARRAY,QARRAY,XARRAY
       COMMON AR,DQ,VS,W
C
C     -----  ENTER THE APPROPRIATE VALUES BELOW ------
C
       N= 101
       GRAPHS=1
C
C     ----- FOR GRAPH = 1 OR 2 MUST ENTER CHANGES IN SUBROUTINE -----
C     ----- BORDER AND SUBROUTINE  EXACT -----
C     ----- LINES "FIRST ORDER   N = ???"
C     -----       "DENSITY RATIO = ???  TEMP RATIO = ???"
C     -----       "PRESSURE RATIO = ???"
C
       SKIP=18
       JSTOP=101
       TRI=1.DDD0
       PRI=5.DD00
       DRI=5.DDD0
       QLI=D.DDDD
       QRI=0.0DDD
       LBDPRI = 1.DD0
       LBDTRI = 1.D00
       LBDDRI = 1.DDD
       RBDDRI = 1.D00
       RBDPRI = 4.DDD/5.D00
       RBDTRI = 1.D00
       G=1.4DD0
       EE=0.1D-8
C
C ----- DENOTE LOW PRESSURE SIDE BY SETTING -----
C ----- LWPRES = 2 IF LOW PRESSURE ON RIGHT
C ----- LWPRES = 3 IF LOW PRESSURE ON LEFT
C
       LWPRES = 3
C
C ----- SET BOUNDARY CONDITIONS BY SPECIFING OPEN OR CLOSED -----
C ----- LBNDRY: OPEN = 0, CLOSED = 1 (FOR LEFT BOUNDARY) -----
C ----- IF OPEN SPECIFY IF PRESSURE IS TO BE MAINTAINED AT LBDPRI -----
C --- OR IF IT CAN ADJUST TO PREVENT ANY WAVES FORMING AT THE BOUNDARY -
C ----- LBDPRS: CONSTANT = 0, ADJUSTABLE = 1 -----
C ----- DO THE SAME FOR RIGHT BOUNDARY; RBNDRY,RBDPRS -----
C
       LBNDRY = 1
       LBDPRS = 1
       RBNDRY = 0
       RBDPRS = D
C     ---------  EXACT  SOLUTION  VALUES  ------------
       XINIT=D.5D
       VHEAD= 1.D
       VTAIL= 0.310557
       VCDE=-.574487
       VSE=-1.4D2346
       DLCD=2.713115
       DLSH=1.69344
       SIGMA(1,2)=D.5DDDDDD1D0D
       SIGMA(2,2)=D.5DD0DDD1DDD
       SIGMA(3,2)=D.5DDDDD01DDD
       SIGMA(4,2)=D.5DDDDDD1DDD
C
C     ++++++++++  END  OF  USER  INPUT  AREA  ++++++++
C
       SK = D
       T=0.0DD0
       DO 1D I=1,4
            I2(I)=0
            LNODE(I) = 0
            RNODE(I) = D
            X2(I)=D.DD0D
```

```
            SIGMA(I,1)=0.0D00
   10 CONTINUE
         ND=DBLE(N)
         H=1.D00/(ND-1.D00)
         DO 11 I=1,N
            A(I)=0.0D00
            Q(I)=0.DD00
            NEWQQ(I)=0.0D00
            NEWRR(I)=0.0D0D
            NEWS(I)=0.0D00
            QARRAY(I)=D.D
            PARRAY(I)=D.0
            DARRAY(I)=0.0
            SARRAY(I)=0.0
            XARRAY(I)=FLOAT(I-1)*SNGL(H)
   11 CONTINUE
         DELT=2.0D00
C
C -----LOAD INITIAL REIMAN VALUES INTO NODE LOCATIONS,FIRST FROM NODE---
C -----1 TO MIDPOINT (Y), AND THEN FROM Y TO N. NOTE IF SHOCK DOES NOT--
C -----START AT MIDPOINT THEN Y SHOULD BE SET TO NODE WHERE SHOCK -----
C -----INITIALLY IS-----
C
         AR=1.0D00
         DQ=D.0D00
         W=1.DD00
         VS=0.0D00
         QCS=0.D00
         VHEW=0.D00
         VTEW=0.D00
         G1=1.D00/(G-1.D00)
         G2=2.D00/(G-1.D00)
C
C ----- FLOW RIGHT -----
C
         IF(LWPRES.EQ.2) THEN
         LBDDR = DRI * LBDDRI
         LBDPR = PRI * LBDPRI
         LBDTR = TRI * LBDTRI
         QLBD = QLI
         RBDDR = RBDDRI
         RBDPR = RBDPRI
         RBDTR = RBDTRI
         QRBD = QRI
            Y = (N+1)/2
C        Y = 38
            DO 12 I=1,Y
               S(I)=G2-(G1/G)*DLOG(PRI/((DRI)**G))
               QQ(I)=QLI+DSQRT(TRI)*S(I)
               RR(I)=QLI-OSQRT(TRI)*S(I)
   12     CONTINUE
            Y=(N+3)/2
C        Y=39
            DO 13 I=Y,N
               S(I)=G2
               QQ(I)=QRI+S(I)
               RR(I)=QRI-S(I)
   13     CONTINUE
         ELSE
C
C ----- FLOW LEFT -----
C
         LBDDR = LBDDRI
         LBDPR = LBDPRI
         LBDTR = LBDTRI
         QLBD = QLI
         RBDDR = RBDDRI * DRI
         RBDPR = RBDPRI * PRI
         RBDTR = RBDTRI * TRI
```

```
         QRBD = QRI
            Y = (N-1)/2
C        Y = 13
            DO 14 I=1,Y
               S(I)=G2
               QQ(I)=QLI+S(I)
               RR(I)=QLI-S(I)
   14    CONTINUE
            Y = (N+1)/2
C        Y = 14
            DO 15 I=Y,N
               S(I)=G2-(G1/G)*DLOG(PRI/((DRI)**G))
               QQ(I)=QRI+DSQRT(TRI)*S(I)
               RR(I)=QRI-DSQRT(TRI)*S(I)
   15    CONTINUE
         END IF
C
C ----- SET UP GRAPHICS PLOTS OF VARIABLES -----
C
         IF (GRAPHS.GT.0) THEN
            CALL COMPRS
C        CALL TEK618
            CALL HWROT('AUTO')
            CALL HWSCAL('SCREEN')
            IF (GRAPHS.EQ.2) THEN
               CALL PAGE(11.0,8.5)
            ELSE
               CALL PAGE(8.5,11.0)
            END IF
            IF (GRAPHS.EQ.1) THEN
            CALL BORDER(JSTOP)
            END IF
         END IF
         HALT = 0
         J=1
         COUNT=1
         IF (GRAPHS.EQ.1) THEN
            CALL PLOT(J,JSTOP,N,QQ,RR,S,H,XARRAY,PARRAY,
        #DARRAY,QARRAY,SARRAY,G,G1,G2)
         END IF
C
C ----- BURST DIAPHRAGM -----
C
         CALL TIME(N,QQ,RR,S,DELT,H)
         CALL DBURST(N,H,QQ,RR,S,G,G1,G2,DELT,I2,X2,W,AR,DQ,VS,LWPRES,
        C              SIGMA,A,Q)
C
         IF (GRAPHS.EQ.0) THEN
            CALL LIST(N,SIGMA,QQ,RR,S,G,G1,G2,J,T,DELT,VS,QCS,VTEW,VHEW,
        C              XINIT,VHEAD,VTAIL,VCDE,VSE)
         END IF
C
C ----- BEGIN CALCULATION FOR JUMP TO NEXT TIME AND CONTINUE -----
C ----- UNTIL EITHER JSTOP REACHED OR SHOCK MEETS CONTACT SURFACE -----
C
   16 IF (J.EQ.JSTOP) GOTO 18
         PLTCNT=J/SKIP
C
         CALL TIME(N,QQ,RR,S,DELT,H)
C
         CALL TRAK(N,SIGMA,H,QQ,RR,S,G,G1,G2,DELT,I2,X2,W,AR,DQ,VS,J,
        C          LWPRES,QCS,VHEW,VTEW)
C
         CALL SWEEP(N,H,SIGMA,QQ,RR,S,DELT,EE,Q,A,NEWQQ,NEWRR,NEWS,I2,G2,
        C          J,LNODE,RNODE,HALT,LBNDRY,LBDPRS,LBDPR,LBDTR,LBDDR,
        C          QLBD,G,G1,RBNDRY,RBDPRS,RBDPR,RBDTR,RBDDR,QRBD,BDRY,
        C          SK)
C
         IF(LNODE(4).LT.J) THEN
```

```
              LNODE(1) = 0
              RNODE(1) = 0
              GO TO 17
        END IF
        IF(RNODE(4).LT.LNODE(4)) THEN
             RNODE(1) = 0
        END IF
C
        CALL CORRCT(LNODE,RNODE,N,SIGMA,H,QQ,RR,S,G,G1,G2,I2,X2,W,AR,DQ,
     C             VS,A,Q)
C
   17 IF (SIGMA(1,2).GE.1.D00) THEN
              IF(RBNDRY.EQ.0) THEN
                     IF(SIGMA(1,2).NE.3.D00) THEN
                     SK = 2
                     CALL BONDRY(Q(N),Q(N-1),QRBD,A(N),A(N-1),QQ(N),QQ(N-1),
     C                     RR(N),RR(N-1),S(N),S(N-1),H,EE,DELT,
     C                     RBNDRY,RBDPRS,RBDPR,RBDDR,RBDTR,J,NEWQQ(N),
     C                     NEWRR(N),NEWS(N),G,G1,G2,HALT,BDRY,SK)
                     END IF
              ELSE
                     CALL EXTRAP(RR(N-1),RR(N-2),QQ(N-1),QQ(N-2),S(N-1),
     C                     S(N-2),H,H,RRA,QQA,SA,AA,QA)
                     CALL SRFLCT(QQA,RRA,SA,SIGMA,VS,DELT,LWPRES,
     C                     RR(N),QQ(N),S(N),Q(N),A(N),G,G1,G2)
              END IF
        END IF
C
        IF (SIGMA(1,2).LE.0.D00) THEN
              IF(LBNDRY.EQ.0) THEN
                     IF(SIGMA(1,2).NE.-2.D00) THEN
                     BDRY = 3
                     SK=2
                     CALL BONDRY(Q(1),Q(2),QLBD,A(1),A(2),QQ(1),QQ(2),
     C                     RR(1),RR(2),S(1),S(2),H,EE,DELT,
     C                     LBNDRY,LBDPRS,LBDPR,LBDDR,LBDTR,J,NEWQQ(1),
     C                     NEWRR(1),NEWS(1),G,G1,G2,HALT,BDRY,SK)
                     END IF
              ELSE
                     CALL EXTRAP(RR(2),RR(3),QQ(2),QQ(3),S(2),
     C                     S(3),H,H,RRB,QQB,SB,AB,QB)
                     CALL SRFLCT(QQB,RRB,SB,SIGMA,VS,DELT,LWPRES,
     C                     RR(1),QQ(1),S(1),Q(1),A(1),G,G1,G2)
              END IF
        END IF
        T=T+DELT
C
C ----- OUTPUT DATA -----
C
        IF ((COUNT.EQ.PLTCNT*SKIP).AND.(GRAPHS.EQ.1))
     C    THEN
C     IF((J.GT.55)) THEN
            CALL PLOT(J,JSTOP,N,QQ,RR,S,H,XARRAY,PARRAY,
     CDARRAY,QARRAY,SARRAY,G,G1,G2)
C     END IF
        END IF
        IF ((COUNT.EQ.PLTCNT*SKIP).AND.(GRAPHS.EQ.0))
     C     THEN
           CALL LIST(N,SIGMA,QQ,RR,S,G,G1,G2,J,T,DELT,VS,QCS,VTEW,VHEW,
     C              XINIT,VHEAD,VTAIL,VCDE,VSE)
        END IF
        IF ((COUNT.EQ.PLTCNT*SKIP).AND.(GRAPHS.EQ.2))
     C    THEN
        IF (J.GT.50) THEN
           CALL EXACT(N,XINIT,T,VHEAD,VTAIL,VCDE,VSE,DLCD,DLSH,QQ,RR,S,H,
     CXARRAY,DARRAY,G,G1,G2,DRI)
        END IF
        END IF
C
```

```
      IF(HALT.EQ.1) GO TO 18
      J=J+1
      COUNT=COUNT+1
      GO TO 16
   18 CALL DONEPL
      END
C     *********************************************
C     *********************************************
C     *********************************************
C
      SUBROUTINE LIST(N,SIGMA,QQ,RR,S,G,G1,G2,J,T,DELT,VS,QCS,VTEW,
     C                VHEW,XINIT,VHEAD,VTAIL,VCDE,VSE)
C
C     ++++++++++++++++++++++++++++++++++++++++++++++
C     +                                            +
C     +       TABULAR  RESULTS  SUBROUTINE         +
C     +                                            +
C     ++++++++++++++++++++++++++++++++++++++++++++++
C
C     ----------- VARIABLE DEFINITIONS ---------------
C
C     DENS  - DENSITY
C     PRESS - PRESSURE
C     TEMP  - TEMPERATURE
C
      INTEGER I,J,N,L
      DIMENSION SIGMA(4,2),QQ(N),RR(N),S(N)
      DOUBLE PRECISION SIGMA,QQ,RR,S,PRESS,VTEW,VHEW,QCS,TEWXE,
     C                 TEMP,DENS,G,G1,G2,Q,T,DELT,VS,HEWXE,
     C                 XINIT,VHEAD,VTAIL,VCDE,VSE,SKXE,CSXE
C
      WRITE(9,*) 'TIME LEVEL',J,'                  ELAPSED TIME IS',T
      WRITE(9,*) 'TIME STEP IS',DELT,'    SHOCK VELOCITY IS',VS
      WRITE(9,*) 'CONTACT SURFACE VELOCITY IS',QCS
      WRITE(9,*) 'HEAD EXPANSION WAVE VELOCITY IS',VHEW
      WRITE(9,*) 'TAIL EXPANSION WAVE VELOCITY IS',VTEW
      WRITE(9,*) ' '
      WRITE(9,*) '   NODE         VELOCITY           DENSITY
     CPRESSURE'
      WRITE(9,*) ' '
      DO 61 I=1,N
      TEMP=(QQ(I)-RR(I))*(QQ(I)-RR(I))/(4.D00*S(I)*S(I))
      DENS=((1.D00/TEMP)*DEXP(G*(1.D00-G)*(S(I)-G2)))**(-G1)
      PRESS=TEMP*DENS
      Q=(QQ(I)+RR(I))/2.0D00
      WRITE (9,65) I,Q,DENS,PRESS
   65 FORMAT (4X,I2,7X,F12.6,7X,F12.6,7X,F12.6)
   61 CONTINUE
      WRITE(9,*) ' '
      WRITE(9,*) ' '
      WRITE(9,*) '   NODE            QQ                  RR
     CMODIFIED S'
      WRITE(9,*) ' '
      DO 62 I=1,N
      WRITE (9,66) I,QQ(I),RR(I),S(I)
   66 FORMAT (4X,I2,7X,F12.6,7X,F12.6,7X,F12.6)
   62 CONTINUE
      WRITE(9,*) ' '
      WRITE(9,*) ' '
      WRITE(9,*) '      DISCONTINUITY LOCATIONS AT TIME LEVEL',J
      WRITE(9,*) ' '
      WRITE(9,*) '        TYPE          LOCATION'
      DO 63 L=1,4
         WRITE(9,*) L,'     ',SIGMA(L,2)
   63 CONTINUE
      WRITE(9,*) ' '
      WRITE(9,*) '-----------------------------------------------------
     C--------------------------------------------------------------'
      WRITE(9,*) ' '
```

```
      WRITE(9,*) ' '
      IF(J.EQ.1) THEN
      WRITE(10,*) '    TIME            SHOCK          CONTACT        HEAD
     C  TAIL'
      WRITE(10,*) '  '
      END IF
      WRITE(10,67) T,SIGMA(1,1),SIGMA(2,1),SIGMA(3,1),SIGMA(4,1)
   67 FORMAT (1X,F12.6,1X,F12.6,1X,F12.6,1X,F12.6,1X,F12.6)
      IF(J.EQ.1) THEN
      WRITE(8,*) '    EXACT VALUES'
      WRITE(8,*) '    TIME            SHOCK          CONTACT        HEAD
     C  TAIL'
      WRITE(8,*) '  '
      END IF
      SKXE=XINIT+VSE*T
      CSXE=XINIT+VCDE*T
      HEWXE=XINIT+VHEAD*T
      TEWXE=XINIT+VTAIL*T
      WRITE(8,68) T,SKXE,CSXE,HEWXE,TEWXE
   68 FORMAT (1X,F12.6,1X,F12.6,1X,F12.6,1X,F12.6,1X,F12.6)
      RETURN
      END
C
      SUBROUTINE TIME(N,QQ,RR,S,DELT,H)
C
C    *******************************************
C    *                                         *
C    *     CALCULATE  TIME  STEP  SUBROUTINE    *
C    *                                         *
C    *******************************************
C
C    --------NEW VARIABLE  DEFINITIONS  ------------
C
C    TMIN - RUNNING VALUE OF THE MINIMUM TIME STEP
C
      INTEGER N,I
      DIMENSION QQ(N),RR(N),S(N)
      DOUBLE PRECISION H,A,QQ,RR,S,DELT,TMIN,Q
      TMIN=2.0D00
      DO 21 I=1,N
           A=(QQ(I)-RR(I))/(2.D00*S(I))
           Q=(QQ(I)+RR(I))/2.0D00
           DELT=H/(DABS(DABS(Q)+A))
           IF (DELT.LT.TMIN) THEN
           TMIN=DELT
           END IF
   21 CONTINUE
      DELT=0.99D00*TMIN
      RETURN
      END
C
      SUBROUTINE TRAK(N,SIGMA,H,QQ,RR,S,G,G1,G2,DELT,I2,X2,W,AR,DQ,VS,
     #J,LWPRES,QCS,VHEW,VTEW)
C
C    *******************************************
C    *                                         *
C    *     DISCONTINUITY TRACKING SUBROUTINE    *
C    *                                         *
C    *******************************************
C
C    -----------  VARIABLE  DEFINITIONS -------------
C
C    CSDIR - CONTACT SURFACE DIRECTION, 2 TO THE RIGHT,3 TO THE LEFT
C    CSRMN - RIEMANN VARIABLE CHANGE ACROSS A CONTACT SURFACE
C    DR    - THE RATIO OF THE DENSITY ACROSS A
C            SHOCK, B/A(RIGHT),B/A(LEFT)
C    DREMN - DUMMY VARIABLE
C    PR    - THE RATIO OF THE PRESSURE ACROSS A
C            SHOCK, A/B(RIGHT),B/A(LEFT)
```

```
C     MREIMN - THE MEASURED JUMP IN QQ ACROSS THE SHOCK,
C              FROM A TO B.
C     EREIMN - THE JUMP IN QQ ACROSS THE SHOCK CALCULATED ANALYTICALLY
C              AS A FUNCTION OF W.
C     SAP - ENTROPY TO THE LEFT OF THE SHOCK FOR FLOWS RIGHT
C           OR ENTROPY TO THE RIGHT OF THE C.S. FOR FLOWS LEFT
C     SBP - ENTROPY TO THE RIGHT OF THE C.S. FOR FLOWS RIGHT
C           OR ENTROPY TO THE LEFT OF THE SHOCK FOR FLOWS LEFT
C     SHKDIR - SHOCK DIRECTION OF TRAVEL,3 TO THE LEFT,2 TO THE RIGHT
C     TS - TIME FOR SHOCK TO TRAVEL ONE INTERVAL
C     X - DISTANCE FROM LEFT BOUNDARY TO NODE
C
        INTEGER N,I,Y,I2,L,J,SHKDIR,CSDIR,LWPRES
        DIMENSION SIGMA(4,2),X2(4),RR(N),QQ(N),S(N),I2(4)
        DOUBLE PRECISION SIGMA,X2,X,H,AB,SA,SB,AA,QA,QB,QQA,QQB,RRA,RRB,
     C                   RR,QQ,S,TS,W,DQ,AR,PR,G,G1,G2,VS,DELT,CSRMN,
     C                   Q,MREIMN,DREMN,EREIMN,WW,SA1,SA2,SAP,SBP
     C                   AW,QW,VHEW,VTEW,TIME,QCS
C
C     ++++++++  LOCATING THE UPSTREAM NODE +++++++++++
C
        DO 10 L=1,4
             SIGMA(L,1)=SIGMA(L,2)
             Y=0
             X=0.D00
             I=1
 11      IF (.NOT.(Y.EQ.0)) GOTO 10
                IF (SIGMA(L,1).LT.X) THEN
                     X2(L)=X
                     I2(L)=I
                     Y=1
                END IF
             X=X+H
             I=I+1
             GOTO 11
 10    CONTINUE
C
C ----- IF SHOCK HAS LEFT AN OPEN BOUNDARY OUT OF THE TUBE THEN SET ----
C ----- SHOCK TO NEUTRAL ------
C
        IF (I2(1).GT.N) THEN
               SIGMA(1,1) = 2.D00
               SIGMA(1,2) = 3.D00
               W = 1.D00
               VS = 1.D00
               DQ = 0.D00
               AR = 0.D00
               PR = 1.D00
               DR = 1.D00
               GO TO 150
        ELSE IF(I2(1).LT.2) THEN
               SIGMA(1,1) =-1.D00
               SIGMA(1,2) =-2.D00
               W = 1.D00
               VS =-1.D00
               DQ = 0.D00
               AR = 0.D00
               PR = 1.D00
               DR = 1.D00
               GO TO 150
        END IF
C
C     +++++++++++++  DETERMINING SHOCK SPEED  ++++++++
C
          IF((J.EQ.1).OR.(I2(1).EQ.2).OR.(I2(1).EQ.N)) THEN
C
C -----AT TIME ZERO OR BOUNDARYS DETERMINE CORRECT SHOCK DIRECTION-----
C -----SHKDIR = 3 IS A SHOCK HEADED LEFT, AND SHKDIR = 2 IS SHOCK-----
C -----TRAVELING RIGHT-----
```

```
C
                IF(LWPRES.EQ.3) THEN
                      SHKDIR = 3
                      IF (J.EQ.1) THEN
                            X2(1) = X2(1) - H
                            I2(1) = I2(1) - 1
                            X2(2) = X2(2) - H
                            I2(2) = I2(2) - 1
                      END IF
                      GO TO 20
                ELSE
                      SHKDIR = 2
                END IF
                GO TO 20
C
C -----IF SHOCK AND CONTACT SURFACE ARE NOT WITHIN THE SAME-----
C -----INTERVAL THEN NO CORRECTIONS ARE NEEDED IN CALCULATING-----
C -----THE REIMAN VARIABLE JUMP ACROSS THE SHOCK-----
C
          ELSE IF (I2(1).NE.I2(2)) THEN
                IF((SHKDIR.EQ.3).AND.(SIGMA(1,1).EQ.(X2(1)-H))) THEN
                      X2(1) = X2(1) - H
                      I2(1) = I2(1) - 1
                END IF
                GO TO 20
C
C -----IF SHOCK AND CONTACT SURFACE ARE WITHIN THE SAME INTERVAL-----
C -----THEN CORRECTIONS ARE REQUIRED TO DETERMINE SHOCK STRENGTH-----
C
          ELSE IF(SHKDIR.EQ.3) THEN
C
C -----SHOCK LOCATION RELATIVE TO THE CONTACT SURFACE FOR A SHOCK-----
C -----HEADED TO THE LEFT DETERMINES THE CORRECT VALUES FOR W AND VS--
C
                IF(SIGMA(1,1).GT.SIGMA(2,1)) THEN
                      GO TO 111
                ELSE IF(SIGMA(1,1).EQ.(X2(1)-H)) THEN
                      X2(1) = X2(1) - H
                      I2(1) = I2(1) - 1
                      X2(2) = X2(2) - H
                      I2(2) = I2(2) - 1
                END IF
   15            RRA=RR(I2(1)-1)
                      RRB=RR(I2(1))
                      QQA=QQ(I2(1)-1)
                      QQB=QQ(I2(1))
                      SA=S(I2(1)-1)
C
                      QA =(QQA+RRA)/2.D00
                      QB =(QQB+RRB)/2.D00
                      AA =(QQA-RRA)/(2.D00*SA)
                      DQ =(QB-QA)/AA
                      W = DSQRT((DQ**2)*(0.36D00)+1.D00) - (DQ*0.6D00)
                      DQ =(-1.D00)*DQ
                      GO TO 110
C
C -----FOR SHOCK HEADED RIGHT THE SAME PROCEDURE FOR CORRECTIONS ARE---
C -----FOLLOWED-----
C
          ELSE IF(SIGMA(1,1).LT.SIGMA(2,1)) THEN
                      GO TO 111
          ELSE
   17            RRA=RR(I2(1)-1)
                      RRB=RR(I2(1))
                      QQA=QQ(I2(1)-1)
                      QQB=QQ(I2(1))
                      SB=S(I2(1))
                      QA =(QQA+RRA)/2.D00
                      QB =(QQB+RRB)/2.D00
```

```
                        AB =(QQB-RRB)/(2.D00*SB)
                        DQ =(QA-QB)/AB
                        W = DSQRT((DQ**2)*(0.36D00)+1.D00) + (DQ*0.6D00)
                        GO TO 110
               END IF
C
C -----WITH NO SHOCK/CONTACT SURFACE INTERACTION THE JUMP IN REIMAN-----
C -----VARIABLES ARE DETERMINED WITHOUT INTERPOLATION OVER THE -----
C -----INTERVAL. MREIMN IS THE MEASURED JUMP. EREIMN IS THE ANALYTICAL--
C -----VALUE-----
C
   20 RRA=RR(I2(1)-1)
      RRB=RR(I2(1))
      QQA=QQ(I2(1)-1)
      QQB=QQ(I2(1))
      SA=S(I2(1)-1)
      SB=S(I2(1))
   21 AB=(QQB-RRB)/(2.D00*SB)
      AA=(QQA-RRA)/(2.D00*SA)
      IF(SHKDIR.EQ.3) THEN
              MREIMN = (RRB-RRA)/AA
              DREMN = DABS(MREIMN)
      ELSE
              MREIMN = (QQA-QQB)/AB
              DREMN = MREIMN
      END IF
C
C -----ITERATE FOR PROPER VALUE OF W USING THE QUADRATIC FIT OF THE-----
C -----REIMAN VARIABLE CHANGE WITH W CURVE.  NOTE LEFT MOVING SHOCKS----
C -----ARE USED IN THESE EQUATIONS SINCE RRB-RRA/AA=-(QQA-QQB/AB)-----
C
  100 WW= (3.0396408D001-((DREMN+2.7574D00)/0.286337D00))
      W=5.513294D00-DSQRT(WW)
      DQ=2.D00*(W*W-1.D00)/(W*(G+1.D00))
      AR=DSQRT(2.D00*(G-1.D00)*(1.D00+((G-1.D00)*W*W/2.D00))*
     C       (G*G2*W*W-1.D00))/((G+1.D00)*W)
      PR=(2.D00*G/(G+1.D00))*W*W-((G-1.D00)/(G+1.D00))
      DR=((G-1.D00)*W*W+2.D00)/((G+1.D00)*W*W)
      EREIMN=DQ+(AR-1.D00)*G2-(AR*G1/G)*DLOG(PR*(DR**G))
      IF (DABS(EREIMN-DABS(MREIMN)).LT.0.1D-5) GO TO 110
      DREMN = (DABS(MREIMN) - EREIMN) + DREMN
      GOTO 100
C
C -----SHOCK VELOCITY DEPENDS ON DIRECTION SHOCK IS TRAVELING -----
C -----LEFT IS < 0, AND RIGHT IS > 0 -----
C
  110 IF (J.EQ.1) THEN
              TIME = 0.D00
              SA1 = (G1/G)*DLOG((2.D00*G*(W**2)-G+1.D00)/(G+1.D00))
              SA2 = G1*DLOG(((G-1.D00)*(W**2)+2)/((G+1.D00)*(W**2)))
              IF(SHKDIR.EQ.2) THEN
                      SAP = SB - SA1 - SA2
                      SBP = SAP
              ELSE
                      SBP = SA - SA1 - SA2
                      SAP = SBP
              END IF
  103         IF(SHKDIR.EQ.2) THEN
                      CSRMN =((DEXP((SBP-SA)/G2))*(SA)-(SBP))*AR
              ELSE
                      CSRMN =((DEXP((SAP-SB)/G2))*(SB)-(SAP))*AR
              END IF
              IF(SHKDIR.EQ.3) THEN
                      MREIMN =((RRB-RRA)/AA)+CSRMN
                      DREMN = DABS(MREIMN)
              ELSE
                      MREIMN =((QQA-QQB)/AB)-CSRMN
                      DREMN = MREIMN
              END IF
```

```
  101      WW=(3.0396408D01-((DREMN+2.7574D00)/0.286337D00))
                W=5.513294D00-DSQRT(WW)
                DQ=2.D00*(W*W-1.D00)/(W*(G+1.D00))
                AR=DSQRT(2.D00*(G-1.D00)*(1.D00+((G-1.D00)*W*W/2.D00))*
     C               (G*G2*W*W-1.D00))/((G+1.D00)*W)
                PR=(2.D00*G/(G+1.D00))*W*W-((G-1.D00)/(G+1.D00))
                DR=((G-1.D00)*W*W+2.D00)/((G+1.D00)*W*W)
                EREIMN=DQ+(AR-1.D00)*G2-(AR*G1/G)*DLOG(PR*(DR**G))
                IF (DABS(EREIMN-DABS(MREIMN)).LT.0.1D-5) GO TO 102
                DREMN = (DABS(MREIMN) - EREIMN) + DREMN
                GOTO 101
  102      SA1 = (G1/G)*DLOG((2.D00*G*(W**2)-G+1.D00)/(G+1.D00))
                SA2 = G1*DLOG(((G-1.D00)*(W**2)+2)/((G+1.D00)*(W**2)))
                IF(SHKDIR.EQ.2) THEN
                       SAP = SB - SA1 - SA2
                ELSE
                       SBP = SA - SA1 - SA2
                END IF
                IF (DABS(SAP-SBP).LT.0.1D-5) GO TO 105
                IF(SHKDIR.EQ.2) THEN
                      SBP = (SAP-SBP) + SBP
                ELSE
                      SAP = (SBP-SAP) + SAP
                END IF
                GO TO 103
          END IF
  105 IF(SHKDIR.EQ.3) THEN
                DQ = (-1.D00)*DQ
                VS = ((RRA+QQA)*0.5D00) - (W*AA)
          ELSE
                VS=(QQB+RRB)*0.5D00+W*AB
          END IF
          TS=H/DABS(VS)
  111 IF (TS.LT.DELT) THEN
              DELT=0.99D00*TS
          END IF
          SIGMA(1,2)=VS*DELT+SIGMA(1,1)
C
C +++++DETERMINE CONTACT SURFACE SPEED+++++++++
C
          IF(J.EQ.1) THEN
                CSDIR = SHKDIR
          END IF
C
C ----- CONTACT SURFACE TRAVELING RIGHT -----
C
  150 IF(CSDIR.EQ.2) THEN
C
C -----CONTACT SURFACE MOVING RIGHT, CHECK FOR SHOCK IN INTERVAL-----
C -----AND CALCULATE SPEED OF CONTACT SURFACE AS APPROPRIATE-----
C
                IF(J.EQ.1) THEN
                       QB =(QQB+RRB)/2.D00
                       QA = DQ*AB + QB
                       CALL SKJUMP(AB,QB,SB,AR,DQ,VS,G,G1,W,AA,QA,SA)
                       Q = QA
                       GO TO 200
                END IF
                IF(X2(2).EQ.X2(1)) THEN
                       IF(SIGMA(2,1).LE.SIGMA(1,1)) THEN
                              Q = (QQ(I2(2)-1) + RR(I2(2)-1)) / 2.D00
                       ELSE
                              Q =(QQ(I2(2)) + RR(I2(2))) / 2.D00
                       END IF
                ELSE
                       QA = (QQ(I2(2)-1) + RR(I2(2)-1)) / 2.D00
                       QB = (QQ(I2(2)) + RR(I2(2))) / 2.D00
                       Q = (QA + QB)/2.D00
                END IF
```

```
      ELSE IF(CSDIR.EQ.3) THEN
C
C ----- CONTACT SURFACE TRAVELING LEFT -----
C
              IF(J.EQ.1) THEN
                     QA =(QQA+RRA)/2.D00
                     QB = DQ*AA + QA
                     CALL SKJUMP(AA,QA,SA,AR,DQ,VS,G,G1,H,AB,QB,SB)
                     Q = QB
                     GO TO 200
              END IF
              IF(X2(2).EQ.X2(1)) THEN
                     IF(SIGMA(2,1).GE.SIGMA(1,1)) THEN
                            Q = (QQ(I2(2)) + RR(I2(2))) / 2.D00
                     ELSE
                            Q = (QQ(I2(2)-1) + RR(I2(2)-1)) / 2.D00
                     END IF
              ELSE IF(SIGMA(2,1).EQ.(X2(2)-H)) THEN
                     X2(2) = X2(2) - H
                     I2(2) = I2(2) - 1
                     Q = (QQ(I2(2))+RR(I2(2))) / 2.D00
              ELSE
                     QA = (QQ(I2(2)-1) + RR(I2(2)-1)) / 2.D00
                     QB = (QQ(I2(2)) + RR(I2(2))) / 2.D00
                     Q = (QA + QB)/2.D00
              END IF
      END IF
  200 SIGMA(2,2) = DELT * Q + SIGMA(2,1)
      QCS=Q
C
C ----- CALCULATE EXPANSION WAVE SPEED ------
C
      TIME = TIME+DELT
      IF(J.EQ.3) THEN
              AW=(QQ((N+1)/2)-RR((N+1)/2))/(2.D00*S((N+1)/2))
              QW=Q
              IF((CSDIR).EQ.2) THEN
                     VHEW=-(AW)
                     VTEW= QW-AW
              ELSE
                     VHEW= AW
                     VTEW= QW+AW
              END IF
      END IF
      IF(J.EQ.3) THEN
              SIGMA(3,2) = SIGMA(3,1) + VHEW*TIME
              SIGMA(4,2) = SIGMA(4,1) + VTEW*TIME
      ELSE IF((CSDIR.EQ.2).AND.(SIGMA(3,1).GT.0.D00)) THEN
              SIGMA(3,2) = SIGMA(3,1) + VHEW*DELT
              SIGMA(4,2) = SIGMA(4,1) + VTEW*DELT
      ELSE IF((CSDIR.EQ.3).AND.(SIGMA(3,1).GT.1.D00)) THEN
              SIGMA(3,2) = SIGMA(3,1) + VHEW*DELT
              SIGMA(4,2) = SIGMA(4,1) + VTEW*DELT
      END IF
      RETURN
      END
C
      SUBROUTINE SWEEP(N,H,SIGMA,QQ,RR,S,DELT,EE,Q,A,NEWQQ,NEWRR,NEWS,
     CI2,G2,J,LNODE,RNODE,HALT,LBNDRY,LBDPRS,LBDPR,LBDTR,LBDDR,QLBD,G,
     CG1,RBNDRY,RBDPRS,RBDPR,RBDTR,RBDDR,QRBD,BDRY,SK)
C
C     ***********************************************
C     *                                             *
C     *       SPACE  SWEEPING  SUBROUTINE           *
C     *                                             *
C     ***********************************************
C
C     ----------- VARIABLE  DEFINITIONS -----------
C
```

```
C      AAVG - AVERAGE SPEED OF SOUND
C      CNTACT - 3 DIGIT VARIABLE DENOTING CONTACT SURFACE
C               LOCATION,DIRECTION OF TRAVEL, AND IF IT CROSSES A NODE
C      DELQQH - CHANGE IN QQ FROM I TO I+1
C      DELQQL - CHANGE IN QQ FROM I-1 TO I
C      DELRRH - CHANGE IN RR FROM I TO I+1
C      DELRRL - CHANGE IN RR FROM I-1 TO I
C      DELSH  - CHANGE IN S FROM I TO I+1
C      DELSL  - CHANGE IN S FROM I-1 TO I
C      DELAH  - CHANGE IN A FROM I TO I+1
C      DELAL  - CHANGE IN A FROM I-1 TO I
C      DELQH  - CHANGE IN Q FROM I TP I+1
C      DELQL  - CHANGE IN Q FROM I-1 TO I
C      DELX   - INTERPOLATION DISTANCE (LMD*DELT)
C      DLTA** - PREFIX WHICH INDICATES THE SPATIAL
C               CHANGE IN ** FOR ONE TIME STEP.
C      INTEG(K)- RESULT OF INTEGRATING Z(K)
C      **INT  - VALUE OF ** INTERPOLATED BETWEEN NODES
C               ON THE CURRENT TIME LEVEL.
C      LXX - NODE DEFINING THE LEFT INTERVAL
C      *PRIM(K)- SUFFIX WHICH INDICATES THE SPATIAL
C               DERIVITIVE OF * AT THE CURRENT TIME LEVEL.
C      RXX - NODE DEFINING THE RIGHT INTERVAL
C      SAVG - AVERAGE ENTROPY
C      SHOCK - 3 DIGIT VARIABLE DENOTING SHOCK
C              LOCATION,DIRECTION OF TRAVEL, AND IF IT CROSSES A NODE
C      **STEP - THE CHANGE IN TIME OF ** AT A NODE
C               USED TO STEP UP TO THE NEXT TIME LEVEL
C      X      - LOCATION IN SPACIAL PLANE (I-1)*H
C      Z(K)   - RIGHT SIDE OF THE K'TH EQUATION.
C
          INTEGER I,RXX,LXX,SHOCK,CNTACT,I2,J,LNODE,RNODE,HALT,
     C         N,LBNDRY,LBDPRS,RBNDRY,RBDPRS,SK,BDRY
          DIMENSION SIGMA(4,2),S(N),Q(N),A(N),I2(4),QINT(3),AINT(3),Z(3),
     C           NEWQQ(N),NEWRR(N),NEWS(N),INTEG(3),APRIM(3),QPRIM(3),
     C           LNODE(4),RNODE(4),AAVG(3),RR(N),QQ(N)
          DOUBLE PRECISION AAVG,SAVG,G2,X,H,SIGMA,QQ,RR,S,Q,A,G,G1,
     C                  DELQQH,DELQQL,DELRRH,DELRRL,DELSH,DELSL,
     C                  DELAH,DELAL,DELQH,DELQL,DELT,
     C                  QINT,AINT,QQINT,RRINT,SINT,EE,
     C                  NEWQQ,NEWRR,NEWS,LBDPR,LBDTR,LBDDR,QLBD,
     C                  RRSTEP,SSTEP,INTEG,Z,DLTAQQ,QQSTEP,
     C                  DLTARR,DLTAS,APRIM,QPRIM,
     C                  RBDPR,RBDTR,RBDDR,QRBD
          COMMON AR,DQ,VS,W
C
C ----- COMPUTE VELOCITY AND SPEED OF SOUND AT EACH NODE -----
C
          DO 10 I= 1,N
                 Q(I) = (QQ(I) + RR(I)) / 2.0D00
                 A(I) = ((QQ(I) - RR(I)) /(2.0D00 * S(I)))
      10 CONTINUE
C
C ----- ADVANCE LEFT BOUNDARY TO NEW TIME STEP -----
C
          BDRY = 3
          IF(I2(1).EQ.2) THEN
                SK = 1
          ELSE IF(SIGMA(1,2).EQ.-2.D00) THEN
                SK = 3
          ELSE
                SK = 0
          END IF
          CALL BONDRY(Q(1),Q(2),QLBD,A(1),A(2),QQ(1),QQ(2),RR(1),RR(2),
     C             S(1),S(2),H,EE,DELT,LBNDRY,LBDPRS,LBDPR,LBDDR,LBDTR,
     C             J,NEWQQ(1),NEWRR(1),NEWS(1),G,G1,G2,HALT,BDRY,SK)
C
C ----- AT EACH NODE FROM 2 TO N-1 DETERMINE THE BEST ALGORITHM TO USE--
C ----- TO ADVANCE THAT NODE TO THE NEXT TIME STEP -----
```

```
C
      I=2
   11 IF(I.EQ.N) GO TO 1200
           X=FLOAT(I-1)*H
           DELQQH = QQ(I+1) - QQ(I)
           DELQQL = QQ(I) - QQ(I-1)
           DELRRH = RR(I+1) - RR(I)
           DELRRL = RR(I) - RR(I-1)
           DELSH = S(I+1) - S(I)
           DELSL = S(I) - S(I-1)
           DELAH = A(I+1) - A(I)
           DELAL = A(I) - A(I-1)
           DELQH = Q(I+1) - Q(I)
           DELQL = Q(I) - Q(I-1)
C
C ----- DEFINE LEFT SECTOR AND RIGHT SECTOR WRT NODE EXAMINED-----
C
        RXX = I + 1
        LXX = I
C
C ----- TEST FOR SHOCK ------
C
        IF (I2(1).EQ.RXX) THEN
              SHOCK = 200
              GO TO 20
        ELSE IF (I2(1).EQ.LXX) THEN
              SHOCK = 300
              GO TO 20
        ELSE
              SHOCK = 100
        END IF
        GO TO 30
C
C ----- DETERMINE DIRECTION SHOCK IS TRAVELING -----
C
   20  IF (SIGMA(1,1).LT.SIGMA(1,2)) THEN
              SHOCK = SHOCK + 20
        ELSE
              SHOCK = SHOCK + 30
        END IF
C
C ----- DETERMINE IF SHOCK CROSSES A NODE IN THIS TIME INTERVAL -----
C
        IF (SHOCK.EQ.220) THEN
              IF (SIGMA(1,2).GE.(X+H)) THEN
                    SHOCK = SHOCK + 1
              ELSE
                    SHOCK = SHOCK + 2
              END IF
        ELSE IF (SHOCK.EQ.230) THEN
              IF (SIGMA(1,2).LE.X) THEN
                    SHOCK = SHOCK + 1
              ELSE
                    SHOCK = SHOCK + 2
              END IF
        ELSE IF (SHOCK.EQ.320) THEN
              IF (SIGMA(1,2).GE.X) THEN
                    SHOCK = SHOCK + 1
              ELSE
                    SHOCK = SHOCK + 2
              END IF
        ELSE IF (SHOCK.EQ.330) THEN
              IF (SIGMA(1,2).LE.(X-H)) THEN
                    SHOCK = SHOCK + 1
              ELSE
                    SHOCK = SHOCK + 2
              END IF
        END IF
C
```

```
C ----- TEST FOR CONTACT SURFACE -----
C
   30  IF (I2(2).EQ.RXX) THEN
                CNTACT = 200
                GO TO 40
         ELSE IF (I2(2).EQ.LXX) THEN
                CNTACT = 300
                GO TO 40
         ELSE
                CNTACT = 100
         END IF
         GO TO 50
C
C -----DETERMINE DIRECTION CONTACT SURFACE IS TRAVELING -----
C
   40  IF (SIGMA(2,1).LT.SIGMA(2,2)) THEN
                CNTACT = CNTACT + 20
         ELSE
                CNTACT = CNTACT + 30
         END IF
C
C ----- DETERMINE IF CONTACT SURFACE CROSSES A NODE DURING THIS TIME ---
C ----- INTERVAL -----
C
         IF (CNTACT.EQ.220) THEN
                IF (SIGMA(2,2).GE.(X+H)) THEN
                       CNTACT = CNTACT + 1
                ELSE
                       CNTACT = CNTACT + 2
                END IF
         ELSE IF (CNTACT.EQ.230) THEN
                IF (SIGMA(2,2).LE.X) THEN
                       CNTACT = CNTACT + 1
                ELSE
                       CNTACT = CNTACT + 2
                END IF
         ELSE IF (CNTACT.EQ.320) THEN
                IF (SIGMA(2,2).GE.X) THEN
                       CNTACT = CNTACT + 1
                ELSE
                       CNTACT = CNTACT + 2
                END IF
         ELSE IF (CNTACT.EQ.330) THEN
                IF (SIGMA(2,2).LE.(X-H)) THEN
                       CNTACT = CNTACT + 1
                ELSE
                       CNTACT = CNTACT + 2
                END IF
         END IF
C
C ---CHECK IF EITHER A SHOCK OR CONTACT SURFACE WITHIN H OF THIS NODE---
C ---DETERMINE PROPER ALGORITHM TO USE FOR CALCULATING EXTENDED REIMAN--
C ---VARIABLE CHANGE ALONG CHARACTERISTICS AT THIS NODE-----
C
   50  IF (SHOCK.EQ.100.OR.CNTACT.EQ.100) THEN
C
C ---NEITHER A SHOCK NOR A CONTACT SURFACE EXIST NEAR NODE---
C
          IF (SHOCK.EQ.100.AND.CNTACT.EQ.100) THEN
C
C ---TEST FOR SUBSONIC OR SUPERSONIC FLOW---
C
                IF (DABS(Q(I)).LT.A(I)) THEN
                       IF(I.EQ.(I2(1)-2)) THEN
                              IF(J.EQ.1) THEN
                                     IF(SIGMA(1,1).LE.SIGMA(1,2)) THEN
                                            GO TO 200
                                     END IF
                              END IF
```

```
                    END IF
                    IF(I.EQ.(I2(1)+1)) THEN
                          IF(J.EQ.1) THEN
                                IF(SIGMA(1,1).GE.SIGMA(1,2)) THEN
                                      GO TO 300
                                END IF
                          END IF
                    END IF
                    GO TO 100
              ELSE
                    IF (SIGMA(2,1).LE.SIGMA(2,2)) THEN
                          GO TO 200
                    ELSE
                          GO TO 300
                    END IF
              END IF
        END IF
C
C ---SHOCK OR CONTACT SURFACE ON LEFT, HEADED RIGHT, NO NODES CROSSED--
C
        IF (SHOCK.EQ.322.OR.CNTACT.EQ.322) THEN
              IF (DABS(Q(I)).LT.A(I)) THEN
                    GO TO 300
              ELSE
                    GO TO 500
              END IF
        END IF
C
C ---SHOCK OR CONTACT SURFACE ON LEFT, HEADED LEFT, NO NODES CROSSED--
C
       IF (SHOCK.EQ.332.OR.CNTACT.EQ.332) THEN
             GO TO 300
       END IF
C
C ---SHOCK OR CONTACT SURFACE ON LEFT, HEADED RIGHT, NODE IS CROSSED---
C
       IF (SHOCK.EQ.321.OR.CNTACT.EQ.321) THEN
             GO TO 400
       END IF
C
C ---SHOCK OR CONTACT SURFACE ON LEFT, HEADED LEFT, NODE IS CROSSED---
C
       IF (SHOCK.EQ.331.OR.CNTACT.EQ.331) THEN
             GO TO 300
       END IF
C
C ---SHOCK OR CONTACT SURFACE ON RIGHT, HEADED RIGHT, NO NODES CROSSED--
C
       IF (SHOCK.EQ.222.OR.CNTACT.EQ.222) THEN
             GO TO 200
       END IF
C
C ---SHOCK OR CONTACT SURFACE ON RIGHT, HEADED LEFT, NO NODES CROSSED--
C
       IF (SHOCK.EQ.232.OR.CNTACT.EQ.232) THEN
             IF (DABS(Q(I)).LT.A(I)) THEN
                   GO TO 200
             ELSE
                   GO TO 500
             END IF
       END IF
C
C ---SHOCK OR CONTACT SURFACE ON RIGHT, HEADED RIGHT, NODE CROSSED---
C
       IF (SHOCK.EQ.221.OR.CNTACT.EQ.221) THEN
             GO TO 200
       END IF
C
C ---SHOCK OR CONTACT SURFACE ON RIGHT, HEADED LEFT, JUMPS NODE---
```

```
C
         GO TO 400
         END IF
C
C ---BRANCH HERE IF SHOCK TO RIGHT OF CONTACT SURFACE-----
C ---DETERMINE PROPER ALGORITHM TO USE FOR CALCULATING EXTENDED REIMAN--
C ---VARIABLE CHANGE ALONG CHARACTERISTICS AT THIS NODE-----
C
         IF (SIGMA(1,1).GT.SIGMA(2,1)) THEN
C
C ---SHOCK ON LEFT, HEADED RIGHT, NO NODE JUMPED---
C
         IF (SHOCK.EQ.322) THEN
               IF (CNTACT.EQ.322) THEN
                     IF (DABS(Q(I)).LT.A(I)) THEN
                           GO TO 300
                     ELSE
                           GO TO 500
                     END IF
               ELSE IF (CNTACT.EQ.332) THEN
                     GO TO 800
               ELSE IF (CNTACT.EQ.331) THEN
                     GO TO 800
               ELSE
                     GO TO 900
               END IF
         END IF
C
C ---SHOCK ON LEFT, HEADED LEFT, NO NODE JUMPED---
C
         IF (SHOCK.EQ.332) THEN
               IF (CNTACT.EQ.322) THEN
                     GO TO 300
               ELSE IF (CNTACT.EQ.332) THEN
                     GO TO 600
               ELSE IF (CNTACT.EQ.331) THEN
                     GO TO 600
               ELSE IF (CNTACT.EQ.321) THEN
                     GO TO 700
               ELSE
                     GO TO 900
               END IF
         END IF
C
C ---SHOCK ON LEFT, HEADED RIGHT, JUMPS NODE---
C
         IF (SHOCK.EQ.321) THEN
               IF (CNTACT.EQ.322.OR.CNTACT.EQ.321) THEN
                     GO TO 400
               ELSE IF (CNTACT.EQ.332.OR.CNTACT.EQ.331) THEN
                     GO TO 800
               ELSE
                     GO TO 900
               END IF
         END IF
C
C ---SHOCK ON LEFT, HEADED LEFT, JUMPS NODE---
C
         IF (SHOCK.EQ.331) THEN
               IF (CNTACT.EQ.322.OR.CNTACT.EQ.321) THEN
                     GO TO 700
               ELSE IF (CNTACT.EQ.332.OR.CNTACT.EQ.331) THEN
                     GO TO 600
               ELSE
                     GO TO 900
               END IF
         END IF
C
C ---SHOCK ON RIGHT, HEADED RIGHT, NO NODE JUMPED---
```

```
C
      IF (SHOCK.EQ.222) THEN
            IF (CNTACT.EQ.222) THEN
                  GO TO 200
            ELSE IF (CNTACT.EQ.232.OR.CNTACT.EQ.231) THEN
                  GO TO 800
            ELSE IF (CNTACT.EQ.321.OR.CNTACT.EQ.322) THEN
                  GO TO 400
            ELSE IF (CNTACT.EQ.332.OR.CNTACT.EQ.331) THEN
                  GO TO 800
            ELSE
                  GO TO 900
            ENO IF
      END IF
C
C ---SHOCK ON RIGHT, HEAOEO LEFT, NO NOOE JUMPEO---
C
      IF (SHOCK.EQ.232) THEN
            IF (CNTACT.EQ.222) THEN
                  IF (SIGMA(1,2).LT.SIGMA(2,2)) THEN
                        GO TO 700
                  ELSE
                        GO TO 200
                  ENO IF
            ELSE IF (CNTACT.EQ.231.OR.CNTACT.EQ.232) THEN
                  GO TO 600
            ELSE IF (CNTACT.EQ.221) THEN
                  GO TO 700
            ELSE IF (CNTACT.EQ.322) THEN
                  GO TO 400
            ELSE IF (CNTACT.EQ.332.OR.CNTACT.EQ.331) THEN
                  GO TO 600
            ELSE IF (SIGMA(1,2).LT.SIGMA(2,2)) THEN
                  GO TO 710
            ELSE
                  GO TO 400
            ENO IF
      ENO IF
C
C ---SHOCK ON RIGHT, HEAOEO RIGHT, JUMPS NOOE---
C
      IF (SHOCK.EQ.221) THEN
            IF (CNTACT.EQ.221.OR.CNTACT.EQ.222) THEN
                  GO TO 200
            ELSE IF (CNTACT.EQ.231.OR.CNTACT.EQ.232) THEN
                  GO TO 800
            ELSE IF (CNTACT.EQ.321.OR.CNTACT.EQ.322) THEN
                  GO TO 400
            ELSE
                  GO TO 800
            ENO IF
      END IF
C
C ---SHOCK ON RIGHT, HEAOEO LEFT, JUMPS NOOE---
C
      IF (CNTACT.EQ.221.OR.CNTACT.EQ.222) THEN
            GO TO 700
      ELSE IF (CNTACT.EQ.231.OR.CNTACT.EQ.232) THEN
            GO TO 600
      ELSE IF (CNTACT.EQ.322) THEN
            IF (SIGMA(1,2).LT.SIGMA(2,2)) THEN
                  GO TO 710
            ELSE
                  GO TO 400
            END IF
      ELSE IF (CNTACT.EQ.321) THEN
            GO TO 710
      ELSE IF (OABS(Q(I)).LT.A(I)) THEN
            GO TO 600
```

```
         ELSE
               GO TO 800
         END IF
C
C ---BRANCH HERE IF SHOCK IS TO LEFT OF CONTACT SURFACE----
C ---DETERMINE PROPER ALGORITHM TO USE FOR CALCULATING EXTENDED REIMAN--
C ---VARIABLE CHANGE ALONG CHARACTERISTICS AT THIS NODE-----
C
         ELSE IF (SIGMA(1,1).LT.SIGMA(2,1)) THEN
C
C ---SHOCK ON LEFT, HEADED RIGHT, NO NODE CROSSED---
C
         IF (SHOCK.EQ.322) THEN
               IF(CNTACT.EQ.322.OR.CNTACT.EQ.321) THEN
                     GO TO 600
               ELSE IF (CNTACT.EQ.332) THEN
                     IF (SIGMA(1,2).GT.SIGMA(2,2)) THEN
                           GO TO 700
                     ELSE
                           GO TO 300
                     END IF
               ELSE IF (CNTACT.EQ.331) THEN
                     GO TO 700
               ELSE IF (CNTACT.EQ.222.OR.CNTACT.EQ.221) THEN
                     GO TO 600
               ELSE IF (CNTACT.EQ.232) THEN
                     GO TO 400
               ELSE IF (SIGMA(1,2).GT.SIGMA(2,2)) THEN
                     GO TO 710
               ELSE
                     GO TO 400
               END IF
         END IF
C
C ---SHOCK ON LEFT, HEADED LEFT, NO NODE CROSSED---
C
         IF (SHOCK.EQ.332) THEN
               IF (CNTACT.EQ.322) THEN
                     GO TO 800
               ELSE IF (CNTACT.EQ.332) THEN
                     GO TO 300
               ELSE IF (CNTACT.EQ.321) THEN
                     GO TO 800
               ELSE IF (CNTACT.EQ.222.OR.CNTACT.EQ.221) THEN
                     GO TO 800
               ELSE IF (CNTACT.EQ.231.OR.CNTACT.EQ.232) THEN
                     GO TO 400
               ELSE
                     GO TO 900
               END IF
         END IF
C
C ---SHOCK ON LEFT, HEADED RIGHT, JUMPS NODE---
C
         IF (SHOCK.EQ.321) THEN
               IF (CNTACT.EQ.322.OR.CNTACT.EQ.321) THEN
                     GO TO 600
               ELSE IF (CNTACT.EQ.332.OR.CNTACT.EQ.331) THEN
                     GO TO 710
               ELSE IF (CNTACT.EQ.222.OR.CNTACT.EQ.221) THEN
                     GO TO 600
               ELSE IF (CNTACT.EQ.231) THEN
                     GO TO 710
               ELSE IF (SIGMA(1,2).GT.SIGMA(2,2)) THEN
                     GO TO 710
               ELSE
                     GO TO 400
               END IF
         END IF
```

```
C
C ---SHOCK ON LEFT, HEADED LEFT, JUMPS NODE---
C
          IF (SHOCK.EQ.331) THEN
                 IF (CNTACT.EQ.322.OR.CNTACT.EQ.321) THEN
                        GO TO 800
                 ELSE IF (CNTACT.EQ.331.OR.CNTACT.EQ.332) THEN
                        GO TO 300
                 ELSE IF (CNTACT.EQ.221.OR.CNTACT.EQ.222) THEN
                        GO TO 800
                 ELSE
                        GO TO 400
                 END IF
          END IF
C
C ---SHOCK ON RIGHT, HEADED RIGHT, NO NODE CROSSED---
C
          IF (SHOCK.EQ.222) THEN
                 IF (CNTACT.EQ.222.OR.CNTACT.EQ.221) THEN
                        GO TO 600
                 ELSE IF (CNTACT.EQ.231) THEN
                        GO TO 710
                 ELSE IF (CNTACT.EQ.232) THEN
                        IF (SIGMA(1,2).GT.SIGMA(2,2)) THEN
                               GO TO 700
                        ELSE
                               GO TO 200
                        END IF
                 ELSE
                        GO TO 900
                 END IF
          END IF
C
C ---SHOCK ON RIGHT, HEADED LEFT, NO NODES CROSSED---
C
          IF (SHOCK.EQ.232) THEN
                 IF (CNTACT.EQ.222.OR.CNTACT.EQ.221) THEN
                        GO TO 800
                 ELSE IF (CNTACT.EQ.232) THEN
                        IF (DABS(Q(I)).LT.A(I)) THEN
                               GO TO 200
                        ELSE
                               GO TO 500
                        END IF
                 ELSE
                        GO TO 900
                 END IF
          END IF
C
C ---SHOCK ON RIGHT, HEADED RIGHT, JUMPS NODE---
C
          IF (SHOCK.EQ.221) THEN
                 IF (CNTACT.EQ.221.OR.CNTACT.EQ.222) THEN
                        GO TO 600
                 ELSE IF (CNTACT.EQ.231) THEN
                        GO TO 710
                 ELSE IF (CNTACT.EQ.232) THEN
                        GO TO 700
                 ELSE
                        GO TO 900
                 END IF
          END IF
C
C ---SHOCK ON RIGHT, HEADED LEFT, JUMPS NODE---
C
          IF (CNTACT.EQ.221.OR.CNTACT.EQ.222) THEN
                 GO TO 800
          ELSE IF (CNTACT.EQ.231.OR.CNTACT.EQ.232) THEN
                 GO TO 400
```

```
          ELSE
                  GO TO 900
          END IF
C
C ---SHOCK AND CONTACT SURFACE ARE AT THE SAME LOCATION AFTER TIME---
C ---ZERO---
C
          ELSE IF((SHOCK.EQ.222).AND.(CNTACT.EQ.222))THEN
                  SHOCK = 321
                  CNTACT = 321
                  GO TO 400
          ELSE IF((SHOCK.EQ.322).AND.(CNTACT.EQ.322))THEN
                  IF(DABS(Q(I)).LT.A(I)) THEN
                          GO TO 300
                  ELSE
                          GO TO 500
                  END IF
          ELSE IF((SHOCK.EQ.332).AND.(CNTACT.EQ.332)) THEN
                  SHOCK = 231
                  CNTACT = 231
                  GO TO 400
          ELSE IF((SHOCK.EQ.232).AND.(CNTACT.EQ.232)) THEN
                  IF(DABS(Q(I)).LT.A(I)) THEN
                          GO TO 200
                  ELSE
                          GO TO 500
                  END IF
          ELSE
                  GO TO 720
          END IF
C
C ---CALL CONDITION SUBROUTINE WHICH CONTAINS ALGORITHM THAT IS---
C ---NUMERICALLY BEST SUITED FOR THE SITUATION AT NODE I---
C
  100 CALL COND1(Q(I),Q(I+1),A(I),A(I+1),RR(I),QQ(I),S(I),DELQQL,
     C           DELRRH,DELSH,DELSL,DELQH,DELAH,OELQL,DELAL,H,EE,
     C           DELT,QQINT,RRINT,SINT,QPRIM,APRIM,AINT,Q(I-1),A(I-1))
        GO TO 1000
C
  200 CALL COND2(Q(I),Q(I-1),A(I),A(I-1),RR(I),QQ(I),S(I),DELQQL,
     C           OELRRL,DELSL,DELQL,DELAL,DELT,H,EE,QQINT,RRINT,SINT,
     C           QPRIM,APRIM,AINT)
        GO TO 1000
C
  300 CALL COND3(Q(I),Q(I+1),A(I),A(I+1),RR(I),QQ(I),S(I),DELQQH,
     C           DELRRH,DELSH,DELQH,DELAH,DELT,H,EE,QQINT,RRINT,SINT,
     C           QPRIM,APRIM,AINT)
        GO TO 1000
C
  400 CALL COND4(I,SHOCK,CNTACT,J,LNOOE,RNODE,QQSTEP,RRSTEP,SSTEP)
        GO TO 1100
C
  500 CALL COND5(Q(I),Q(I-1),Q(I+1),A(I),A(I-1),A(I+1),RR(I),QQ(I),
     C           S(I),DELQQL,DELQQH,DELRRL,DELRRH,DELSL,DELSH,OELQL,
     C           DELQH,DELAL,DELAH,H,EE,DELT,QQINT,RRINT,SINT,AINT,
     C           QPRIM,APRIM,SHOCK,CNTACT)
        GO TO 1000
C
  600 CALL COND6(SHOCK,CNTACT,HALT)
        QQSTEP=0.00
        RRSTEP=0.DO
        SSTEP=0.DO
        GO TO 1100
C
  700 CALL CON07(SHOCK,CNTACT,DELT,SIGMA,I2,I,H,HALT,Q(I))
        QQSTEP=0.00
        RRSTEP=0.DO
        SSTEP=0'.DO
        GO TO 1100
```

```
C
  710 CALL COND7N(SHOCK,CNTACT,DELT,SIGMA,I2,I,H,HALT,Q(I),X)
        QQSTEP=0.D0
        RRSTEP=0.D0
         SSTEP=0.D0
        GO TO 1100
C
  720 CALL COND7S(SIGMA,HALT,SHOCK,CNTACT)
        QQSTEP=0.D0
        RRSTEP=0.D0
         SSTEP=0.D0
        GO TO 1100
C
  800 CALL COND8(SHOCK,CNTACT,HALT)
        QQSTEP=0.D0
        RRSTEP=0.D0
         SSTEP=0.D0
        GO TO 1100
C
  900 PRINT * ,'AN IMPOSSIBLE SITUATION EXISTS - ERROR'
        QQSTEP=0.D0
        RRSTEP=0.D0
         SSTEP=0.D0
        HALT = 1
        GOTO 1100
C
C     -------- CALCULATE DLTA QQ, DLTA RR & DLTA S
C
 1000   DLTAQQ=QQINT-QQ(I)
           DLTARR=RRINT-RR(I)
           DLTAS=SINT-S(I)
C
C      ------  CALCULATE  Z(K)'S -----------
C
C
           AAVG(1)=(AINT(1)+A(I))/2.0D00
           AAVG(3)=(AINT(3)+A(I))/2.0D00
           AAVG(2)=0.0D00
           SAVG = (SINT+S(I))/2.D00
           Z(1)=-(1.0D00/G2)*AAVG(1)*(SAVG-G2)*(QPRIM(1)-G2*APRIM(1))
           Z(3)=(1.0D00/G2)*AAVG(3)*(SAVG-G2)*(QPRIM(3)+G2*APRIM(3))
           Z(2)=0.0D00
C
C      ----- INTEGRATE THE Z(K)'S ------------
C
           INTEG(2)=0.0D00
           INTEG(1)=Z(1)*DELT
           INTEG(3)=Z(3)*DELT
C
C      -------- SOLVE THE EQUATION -----------
C
           QQSTEP=DLTAQQ+INTEG(1)
           RRSTEP=DLTARR+INTEG(3)
           SSTEP=DLTAS+INTEG(2)
C
C      -------- STORE THE SOLUTION -----------
C
 1100   NEWQQ(I)=QQ(I)+QQSTEP
           NEWRR(I)=RR(I)+RRSTEP
           NEWS(I)=S(I)+SSTEP
C
C      --------- GO TO NEXT NODE  -------------
C
           I=I+1
        GOTO 11
 1200 CONTINUE
        BDRY = 2
        IF(I2(1).EQ.N) THEN
             SK = 1
```

```
        ELSE IF(SIGMA(1,2).EQ.3.D00) THEN
             SK = 3
        ELSE
             SK = 0
        END IF
        CALL BONDRY(Q(N),Q(N-1),QRBD,A(N),A(N-1),QQ(N),QQ(N-1),RR(N),
     C      RR(N-1),S(N),S(N-1),H,EE,DELT,RBNDRY,RBDPRS,RBDPR,RBDDR,
     C       RBDTR,J,NEWQQ(N),NEWRR(N),NEWS(N),G,G1,G2,HALT,BDRY,SK)
C
C     --------- UPDATE THE VARIABLES  ----------
C
        I=1
 1210 IF (I.EQ.N+1) GOTO 1220
            RR(I)=NEWRR(I)
            QQ(I)=NEWQQ(I)
            S(I)=NEWS(I)
        I=I+1
        GOTO 1210
 1220 CONTINUE
        RETURN
        END
C
C       ****************************************************
C       *                                                  *
C       *            CONDITION SUBROUTINES 1-8             *
C       *                                                  *
C       ****************************************************
C
        SUBROUTINE COND1(QI,QIP1,AI,AIP1,RR,QQ,S,DELQQL,DELRRH,DELSH,
     C                  DELSL,DELQH,DELAH,DELQL,DELAL,H,EE,DELT,QQINT,
     C                  RRINT,SINT,QPRIM,APRIM,AINT,QIM1,AIM1)
C
C       ****************************************************
C       *                                                  *
C       *               SUBROUTINE CONDITION 1             *
C       *                                                  *
C       ****************************************************
C
C ----- USED WHEN NO CONTACT SURFACES NOR SHOCKS WITHIN H OF NODE-----
C ----- AND FLOW IS SUBSONIC-----
C ----- CALCULATES QQINT,RRINT,SINT,QPRIM,APRIM-----
C ----- USES FORWARD AND BACKWARD DIFFERENCE SCHEMES-----
C
C
C     ------------ VARIABLE  DEFINITIONS ------------
C
C     AI - A(I)
C     AIM1 - A(I-1)
C     AIP1 _ A(I+1)
C     E(K)   - ACTUAL ERROR IN CHARACTERISTIC SLOPE  CALCULATION.
C     LMD    - SLOPE OF THE CHARACTERISTICS (Q+A,Q,Q-A)
C     QI - Q(I)
C     QIM1 - Q(I-1)
C     QIP1 - Q(I+1)
C
        DIMENSION LMD(3),DELX(3),QINT(3),AINT(3),E(3),QPRIM(3),APRIM(3)
        INTEGER K
        DOUBLE PRECISION QI,QIP1,AI,AIP1,RR,QQ,S,AIM1,QIM1,
     C                  DELQQL,DELRRH,DELSH,DELSL,DELQH,DELAH,DELQL,
     C                  DELAL,H,EE,DELT,LMD,DELX,QINT,AINT,E,
     C                  QQINT,RRINT,SINT,QPRIM,APRIM
C
C   -----INITIAL ESTIMATE OF CHARACTERISTIC SLOPES-----
C
        LMD(1) = QIM1 + AIM1
        LMD(2) = QI
        LMD(3) = QIP1 - AIP1
C
C   -----CALCULATE LINEARLY INTERPOLATED VALUES OF Q AND A-----
```

```
C
      10 K = 1
      20 IF (K.LT.4) THEN
                   DELX(K) = DELT * LMD(K)
                   IF (LMD(K).LT.0.0D00) THEN
                          QINT(K) = QI - (DELX(K) * DELQH / H)
                          AINT(K) = AI - (DELX(K) * DELAH / H)
                   ELSE
                          QINT(K) = QI - (DELX(K) * DELQL / H)
                          AINT(K) = AI - (DELX(K) * DELAL / H)
                   END IF
                   K = K + 1
                   GO TO 20
            END IF
C
C     -----CALCULATE ERROR BETWEEN ESTIMATED SLOPE AND NEW SLOPE-----
C     -----FROM NEW INTERPOLATED VALUES-----
C
            E(1) = DABS(LMD(1) - (QINT(1) + AINT(1)))
            E(2) = DABS(LMD(2) - (QINT(2)))
            E(3) = DABS(LMD(3) - (QINT(3) - AINT(3)))
C
            LMD(1) = QINT(1) + AINT(1)
            LMD(2) = QINT(2)
            LMD(3) = QINT(3) - AINT(3)
C
C     -----COMPARE ERROR TO ERROR TOLERANCE LEVEL, ITERATE TIL MET-----
C
            IF ((E(1).GT.EE).OR.(E(2).GT.EE).OR.(E(3).GT.EE)) GO TO 10
C
C     -----CALCULATE LINEARLY INTERPOLATED VALUES OF REIMAN-----
C     -----VARIABLES AND MODIFIED ENTROPY AT POINT A-----
C
            QQINT = QQ - (DELX(1) * (DELQQL / H))
            IF (LMD(2).LE.0.0D00) THEN
                    SINT = S - (DELX(2) * (DELSH / H))
            ELSE
                    SINT = S - (DELX(2) * (DELSL / H))
            END IF
            RRINT = RR - (DELX(3) * (DELRRH / H))
C
C --- CALCULATE SPATIAL DERIVATIVES ---
C
            QPRIM(1) = (QI-QIM1)/H
            QPRIM(2) = 0.D00
            QPRIM(3) = (QIP1-QI)/H
            APRIM(1) = (AI-AIM1)/H
            APRIM(2) = 0.D00
            APRIM(3) = (AIP1-AI)/H
            RETURN
            END
C
            SUBROUTINE COND2(QI,QIM1,AI,AIM1,RR,QQ,S,DELQQL,DELRRL,DELSL,
          C                 DELQL,DELAL,DELT,H,EE,QQINT,RRINT,SINT,QPRIM,
          C                 APRIM,AINT)
C
C           ******************************************************
C           *                                                    *
C           *              SUBROUTINE CONDITION 2                *
C           *                                                    *
C           ******************************************************
C
C --- BACKWARD DIFFERENCE ALGORITHM FOR CALCULATING ---
C --- RRINT,QQINT,SINT,APRIM,QPRIM,AINT ---
C
            DIMENSION LMD(3),DELX(3),QINT(3),AINT(3),E(3),QPRIM(3),APRIM(3)
            INTEGER K
            DOUBLE PRECISION QI,QIM1,AI,AIM1,RR,QQ,S,DELQQL,DELRRL,DELSL,
          C                 DELQL,DELAL,DELT,H,EE,AINT,QINT,LMD,DELX,E,
```

```
      C                   QQINT,RRINT,SINT,QPRIM,APRIM
C
C   -----INITIAL ESTIMATE OF CHARACTERISTIC SLOPES-----
C
        LMD(1) = QIM1 + AIM1
        LMD(2) = QI
        LMD(3) = QI - AI
C
C   -----CALCULATE LINEARLY INTERPOLATED VALUES OF Q AND A-----
C
     10 K = 1
     20 IF (K.LT.4) THEN
                DELX(K) = DELT * LMD(K)
                QINT(K) = QI - (DELX(K) * DELQL / H)
                AINT(K) = AI - (DELX(K) * DELAL / H)
                K = K + 1
                GO TO 20
        END IF
C
C   -----CALCULATE ERROR BETWEEN ESTIMATED SLOPE AND NEW SLOPE-----
C   -----FROM NEW INTERPOLATED VALUES-----
C
        E(1) = DABS(LMD(1) - (QINT(1) + AINT(1)))
        E(2) = DABS(LMD(2) - QINT(2))
        E(3) = DABS(LMD(3) - (QINT(3) - AINT(3)))
C
        LMD(1) = QINT(1) + AINT(1)
        LMD(2) = QINT(2)
        LMD(3) = QINT(3) - AINT(3)
C
C   -----COMPARE ERROR TO ERROR TOLERANCE LEVEL, ITERATE TIL MET-----
C
        IF ((E(1).GT.EE).OR.(E(2).GT.EE).OR.(E(3).GT.EE)) GO TO 10
C
C   -----CALCULATE LINEARLY INTERPOLATED VALUES OF REIMAN-----
C   -----VARIABLES AND MODIFIED ENTROPY AT POINT A-----
C
        QQINT = QQ - (DELX(1) * (DELQQL / H))
        SINT = S - (DELX(2) * (DELSL / H))
        RRINT = RR - (DELX(3) * (DELRRL / H))
C
C --- CALCULATE SPATIAL DERIVATIVES ---
C
        QPRIM(1) = (QI-QIM1)/H
        QPRIM(2) = 0.D00
        QPRIM(3) = (QI-QIM1)/H
        APRIM(1) = (AI-AIM1)/H
        APRIM(2) = 0.D00
        APRIM(3) = (AI-AIM1)/H
        RETURN
        END
C
        SUBROUTINE COND3(QI,QIP1,AI,AIP1,RR,QQ,S,DELQQH,DELRRH,DELSH,
      C                 DELQH,DELAH,DELT,H,EE,QQINT,RRINT,SINT,QPRIM,
      C                 APRIM,AINT)
C
C       ****************************************************
C       *                                                  *
C       *              SUBROUTINE CONDITION 3              *
C       *                                                  *
C       ****************************************************
C
C --- FORWARD DIFFERENCE ALGORITHM FOR CALCULATING ---
C --- RRINT,QQINT,SINT,APRIM,QPRIM,AINT ---
C
        DIMENSION LMD(3),DELX(3),QINT(3),AINT(3),E(3),QPRIM(3),APRIM(3)
        INTEGER K
        DOUBLE PRECISION QI,QIP1,AI,AIP1,RR,QQ,S,
      C                 DELQQH,DELRRH,DELSH,DELQH,DELAH,
```

```
     C                    DELT,H,EE,AINT,QINT,LMD,DELX,E,
     C                    QQINT,RRINT,SINT,QPRIM,APRIM
C
C   -----INITIAL ESTIMATE OF CHARACTERISTIC SLOPES-----
C
      LMD(1) = QI + AI
      LMD(2) = QI
      LMD(3) = QIP1 - AIP1
C
C   -----CALCULATE LINEARLY INTERPOLATED VALUES OF Q AND A-----
C
   10 K = 1
   20 IF (K.LT.4) THEN
            DELX(K) = DELT * LMD(K)
            QINT(K) = QI - (DELX(K) * DELQH / H)
            AINT(K) = AI - (DELX(K) * DELAH / H)
            K = K + 1
            GO TO 20
      END IF
C
C   -----CALCULATE ERROR BETWEEN ESTIMATED SLOPE AND NEW SLOPE-----
C   -----FROM NEW INTERPOLATED VALUES-----
C
      E(1) = DABS(LMD(1) - (QINT(1) + AINT(1)))
      E(2) = DABS(LMD(2) - QINT(2))
      E(3) = DABS(LMD(3) - (QINT(3) - AINT(3)))
C
      LMD(1) = QINT(1) + AINT(1)
      LMD(2) = QINT(2)
      LMD(3) = QINT(3) - AINT(3)
C
C   -----COMPARE ERROR TO ERROR TOLERANCE LEVEL, ITERATE TIL MET-----
C
      IF((E(1).GT.EE).OR.(E(2).GT.EE).OR.(E(3).GT.EE)) GO TO 10
C
C   -----CALCULATE LINEARLY INTERPOLATED VALUES OF REIMAN-----
C   -----VARIABLES AND MODIFIED ENTROPY AT POINT A-----
C
      QQINT = QQ - (DELX(1) * DELQQH / H)
      SINT = S - (DELX(2) * DELSH / H)
      RRINT = RR - (DELX(3) * DELRRH / H)
C
C --- CALCULATE SPATIAL DERIVATIVES ---
C
      QPRIM(1) = (QIP1-QI)/H
      QPRIM(2) = 0.D00
      QPRIM(3) = (QIP1-QI)/H
      APRIM(1) = (AIP1-AI)/H
      APRIM(2) = 0.D00
      APRIM(3) = (AIP1-AI)/H
      RETURN
      END
C
      SUBROUTINE COND4(I,SHOCK,CNTACT,J,LNODE,RNODE,QQSTEP,RRSTEP,
     C                  SSTEP)
C
C     ******************************************************
C     *                                                    *
C     *              SUBROUTINE CONDITION 4                *
C     *                                                    *
C     ******************************************************
C
C --- A SITUATION EXISTS AT NODE I THAT WILL BE CORRECTED IN ---
C --- SUBROUTINE CORRCT.  *NODE ARRAYS ARE GIVEN INFORMATION ---
C --- ON NODE LOCATION AND SHOCK/CONTACT SURFACE PICTURE ----
C
C    ------------ VARIABLE  DEFINITIONS ------------
C
C    CD4TRK - DENOTES HOW MANY NODES NEED TO BE CORRECTED
```

```
C              THIS TIME STEP
C
        DIMENSION LNODE(4),RNODE(4)
        INTEGER LNODE,RNODE,I,SHOCK,CNTACT,J,CD4TRK
        DOUBLE PRECISION QQSTEP,RRSTEP,SSTEP
C -----ASSIGN FIRST NODE ENCOUNTERED DURING THE NEW TIME STEP TO ----
C -----LNODE -----
C
        IF(LNODE(4).LT.J) THEN
            CD4TRK = 1
        END IF
        IF(CD4TRK.EQ.1) THEN
               LNODE(1) = I
               LNODE(2) = SHOCK
               LNODE(3) = CNTACT
               LNODE(4) = J
C
C -----IF A SECOND NODE WITH CONDITION 4 IS ENCOUNTERED IN THE -----
C -----SWEEP, THIS NODE IS ASSIGNED TO RNODE -----
C
        ELSE
               RNODE(1) = I
               RNODE(2) = SHOCK
               RNODE(3) = CNTACT
               RNODE(4) = J
        END IF
C
C -----LNODE AND RNODE WILL BE "JUMPED" OVER DURING SWEEP THUS-----
C -----THEIR **STEP VALUES ARE SET TO 0-----
C
        QQSTEP = 0.D00
        RRSTEP = 0.D00
         SSTEP = 0.D00
C
        IF(CD4TRK.EQ.1) THEN
               CD4TRK = CD4TRK + 1
        ELSE
               CD4TRK = 1
        END IF
        RETURN
        END
C
        SUBROUTINE COND5(QI,QIM1,QIP1,AI,AIM1,AIP1,RR,QQ,S,DELQQL,
     C                  DELQQH,DELRRL,DELRRH,DELSL,DELSH,DELQL,DELQH,
     C                  DELAL,DELAH,H,EE,DELT,QQINT,RRINT,SINT,AINT,
     C                  QPRIM,APRIM,SHOCK,CNTACT)
C
C       *******************************************************
C       *                                                     *
C       *                SUBROUTINE CONDITION 5               *
C       *                                                     *
C       *******************************************************
C
C --- FOR SUPERSONIC FLOW WITH A DISCONTINUITY ON ONE SIDE OF THE NODE-
C --- CALCULATES QQINT,RRINT,SINT,APRIM,QPRIM,AINT ---
C
        DIMENSION LMD(3),DELX(3),QINT(3),AINT(3),E(3),QPRIM(3),APRIM(3)
        INTEGER K,SHOCK,CNTACT
        DOUBLE PRECISION QI,AI,RR,QQ,S,QIM1,QIP1,AIM1,AIP1,
     C                  EE,DELT,LMD,DELX,QINT,AINT,E,DELQL,DELQH,DELAL,
     C                  QQINT,SINT,RRINT,QPRIM,APRIM,DELAH,H,
     C                  DELQQL,DELQQH,DELRRL,DELRRH,DELSL,DELSH
C
C --- DISCONTINUITY ON LEFT, HEADED RIGHT, NOT CROSSING NODE ---
C
        IF((SHOCK.EQ.322).OR.(CNTACT.EQ.322)) THEN
C
C   -----INITIAL ESTIMATE OF CHARACTERISTIC SLOPES-----.
C
```

```
          LMD(1) = QI + AI
          LMD(2) = QI
          LMD(3) = QIP1 - AIP1
C
C    -----CALCULATE LINEARLY INTERPOLATED VALUES OF Q AND A-----
C
     10 K = 1
     20 IF (K.LT.4) THEN
                DELX(K) = DELT * LMD(K)
                QINT(K) = QI - (DELX(K) * DELQH / H)
                AINT(K) = AI - (DELX(K) * DELAH / H)
                K = K + 1
                GO TO 20
          END IF
C
C    -----CALCULATE ERROR BETWEEN ESTIMATED SLOPE AND NEW SLOPE-----
C    -----FROM NEW INTERPOLATED VALUES-----
C
          E(1) = DABS(LMD(1) - (QINT(1) + AINT(1)))
          E(2) = DABS(LMD(2) - QINT(2))
          E(3) = DABS(LMD(3) - (QINT(3) - AINT(3)))
C
          LMD(1) = QINT(1) + AINT(1)
          LMD(2) = QINT(2)
          LMD(3) = QINT(3) - AINT(3)
C
C    -----COMPARE ERROR TO ERROR TOLERANCE LEVEL, ITERATE TO MEET-----
C
          IF((E(1).GT.EE).OR.(E(2).GT.EE).OR.(E(3).GT.EE)) GO TO 10
C
C    -----CALCULATE LINEARLY INTERPOLATED VALUES OF REIMAN-----
C    -----VARIABLES AND MODIFIED ENTROPY AT POINT A-----
C
          QQINT = QQ - (DELX(1) * DELQQH / H)
          SINT = S - (DELX(2) * DELSH / H)
          RRINT = RR - (DELX(3) * DELRRH / H)
C
C --- CALCULATE SPATIAL DERIVATIVES ---
C
          QPRIM(1) = (QIP1-QI)/H
          QPRIM(2) = 0.D00
          QPRIM(3) = (QIP1-QI)/H
          APRIM(1) = (AIP1-AI)/H
          APRIM(2) = 0.D00
          APRIM(3) = (AIP1-AI)/H
          END IF
C
C --- DISCONTINUITY ON RIGHT, HEADED LEFT, NOT CROSSING NODE ---
C
          IF((SHOCK.EQ.232).OR.(CNTACT.EQ.232)) THEN
C
C    -----INITIAL ESTIMATE OF CHARACTERISTIC SLOPES-----
C
          LMD(1) = QIM1 + AIM1
          LMD(2) = QI
          LMD(3) = QI - AI
C
C    -----CALCULATE LINEARLY INTERPOLATED VALUES OF Q AND A-----
C
     30 K = 1
     40 IF (K.LT.4) THEN
                DELX(K) = DELT * LMD(K)
                QINT(K) = QI - (DELX(K) * DELQL / H)
                AINT(K) = AI - (DELX(K) * DELAL / H)
                K = K + 1
                GO TO 40
          END IF
C
C    -----CALCULATE ERROR BETWEEN ESTIMATED SLOPE AND NEW SLOPE-----
```

```
C    -----FROM NEW INTERPOLATED VALUES-----
C
          E(1) = DABS(LMD(1) - (QINT(1) + AINT(1)))
          E(2) = DABS(LMD(2) - QINT(2))
          E(3) = DABS(LMD(3) - (QINT(3) - AINT(3)))
C
          LMD(1) = QINT(1) + AINT(1)
          LMD(2) = QINT(2)
          LMD(3) = QINT(3) - AINT(3)
C
C    -----COMPARE ERROR TO ERROR TOLERANCE LEVEL, ITERATE TO MEET-----
C
          IF ((E(1).GT.EE).OR.(E(2).GT.EE).OR.(E(3).GT.EE)) GO TO 30
C
C    -----CALCULATE LINEARLY INTERPOLATED VALUES OF REIMAN-----
C    -----VARIABLES AND MODIFIED ENTROPY AT POINT A-----
C
          QQINT = QQ - (DELX(1) * (DELQQL / H))
          SINT = S - (DELX(2) * (DELSL / H))
          RRINT = RR - (DELX(3) * (DELRRL / H))
C
C --- CALCULATE SPATIAL DERIVATIVES ---
C
          QPRIM(1) = (QI-QIM1)/H
          QPRIM(2) = 0.D00
          QPRIM(3) = (QI-QIM1)/H
          APRIM(1) = (AI-AIM1)/H
          APRIM(2) = 0.D00
          APRIM(3) = (AI-AIM1)/H
          END IF
          RETURN
          END
C
          SUBROUTINE COND6(SHOCK,CNTACT,HALT)
C
C         *****************************************************
C         *                                                   *
C         *                SUBROUTINE CONDITION 6             *
C         *                                                   *
C         *****************************************************
C
C
          INTEGER SHOCK,CNTACT,HALT
C
          PRINT * ,'THE SITUATION FOR CONDITION 6 WITH THE CONTACT SURFACE '
          PRINT * ,'TO THE RIGHT OF A SHOCK BOTH HEADED RIGHT OR THE       '
          PRINT * ,'CONTACT SURFACE TO THE LEFT OF A SHOCK BOTH HEADED LEFT'
          PRINT * ,'ADDITIONAL LOGIC IS REQUIRED TO PROCEED'
          PRINT * ,'SHOCK =',SHOCK,'  CONTACT SURFACE =',CNTACT
          HALT = 1
          RETURN
          END
C
          SUBROUTINE COND7(SHOCK,CNTACT,DELT,SIGMA,I2,I,H,HALT,Q)
C
C         *****************************************************
C         *                                                   *
C         *                SUBROUTINE CONDITION 7             *
C         *                                                   *
C         *****************************************************
C
C
          DIMENSION SIGMA(4,2),I2(4)
          INTEGER HALT,SHOCK,CNTACT,I2,I
          DOUBLE PRECISION DELT,SIGMA,H,Q
C
          PRINT * ,'CONDITION 7 REQUIRES THAT THE CONTACT SURFACE AND     '
          PRINT * ,'SHOCK(MOVING IN OPPOSITE DIRECTIONS) MEET AND CROSS.  '
          PRINT * ,'WHEN THEY INTERSECT,THE RESULT IS A FUNCTION OF EXIST-'
```

```
      PRINT * ,'ING CONDITIONS AROUND THEM.  BOTH THE ORIGINAL SHOCK  '
      PRINT * ,'AND CONTACT SURFACE WILL EXPERIENCE VELOCITY CHANGES .'
      PRINT * ,'THIS SUBROUTINE WOULD HAVE TO CALCULATE WHEN AND WHERE'
      PRINT * ,'WITHIN THE TIME/SPACE ITERVAL THE SHOCK AND CONTACT   '
      PRINT * ,'SURFACE INTERSECT.  THIS IS BASED ON KNOWN SPEED AND  '
      PRINT * ,'THEIR RESPECTIVE LOCATIONS.  THIS NEW TIME, DELT(NEW) '
      PRINT * ,'THEN COULD BE USED TO RERUN "SWEEP", WITH CONDITION 7S'
      PRINT * ,'EXISTING AT TIME = T + DELT(NEW).  ADDITIONAL LOGIC IS'
      PRINT * ,'REQUIRED TO CONTINUE.'
      PRINT * ,'SHOCK =',SHOCK,'  CONTACT SURFACE =',CNTACT
      PRINT * ,'SIGMA(1,1) =',SIGMA(1,1),'  SIGMA(1,2) =',SIGMA(1,2)
      PRINT * ,'SIGMA(2,1) =',SIGMA(2,1),'  SIGMA(2,2) =',SIGMA(2,2)
      PRINT * ,'CONTACT SURFACE VELOCITY =',Q
      PRINT * ,'NODE I =',I
      HALT = 1
      RETURN
      END
C
      SUBROUTINE COND7N(SHOCK,CNTACT,DELT,SIGMA,I2,I,H,HALT,Q,X)
C
C     ******************************************************
C     *                                                    *
C     *              SUBROUTINE CONDITION 7N               *
C     *                                                    *
C     ******************************************************
C
      DIMENSION SIGMA(4,2),I2(4)
      INTEGER HALT,SHOCK,CNTACT,I2,I
      DOUBLE PRECISION DELT,SIGMA,H,Q,X
C
      PRINT * ,'CONDITION 7N REQUIRES THAT WHEN EITHER A SHOCK OR    '
      PRINT * ,'CONTACT SURFACE JUMPS A NODE (AS DETERMINED BY       '
      PRINT * ,'COMPARING SIGMA(L,1) TO SIGMA(L,2)) A CONTACT SURFACE'
      PRINT * ,'OR SHOCK WOULD BE MET AND CROSSED DURING THE JUMP.   '
      PRINT * ,'THE RESULT WHEN THEY INTERSECT IS A FUNCTION OF      '
      PRINT * ,'EXISTING CONDITIONS AROUND THEM.  BOTH THE ORIGINAL'
      PRINT * ,'SHOCK AND THE CONTACT SURFACE WILL EXPERIENCE VELOCITY'
      PRINT * ,'CHANGES.  THIS SUBROUTINE WOULD HAVE TO CALCULATE WHEN'
      PRINT * ,'WITHIN THE TIME INTERVAL AND WHERE SPACIALLY THE      '
      PRINT * ,'INTERSECTION OCCURS.  THIS IS BASED ON KNOWN SPEED AND'
      PRINT * ,'THE RESPECTIVE LOCATIONS OF THE SHOCK AND CONTACT     '
      PRINT * ,'SURFACE.  THE NEW TIME, DELT(NEW), COULD THEN BE USED '
      PRINT * ,'TO RERUN "SWEEP" WITH CONDITION 4 AT THIS NODE, AND   '
      PRINT * ,'SIGMA(2,2) = SIGMA(1,2) SO THAT CONDITION 7S WOULD    '
      PRINT * ,'RESULT IN THE NEXT TIME INTERVAL.'
      PRINT * ,'SHOCK =',SHOCK,'CONTACT SURFACE =',CNTACT
      PRINT * ,'SIGMA(1,1) =',SIGMA(1,1),'  SIGMA(1,2) =',SIGMA(1,2)
      PRINT * ,'SIGMA(2,1) =',SIGMA(2,1),'  SIGMA(2,2) =',SIGMA(2,2)
      PRINT * ,'CONTACT SURFACE VELOCITY =',Q
      PRINT * ,'NODE I =',I,'  LOCATION AT X =',X
      HALT = 1
      RETURN
      END
C
      SUBROUTINE COND7S(SIGMA,HALT,SHOCK,CNTACT)
C
C     ******************************************************
C     *                                                    *
C     *              SUBROUTINE CONDITION 7S               *
C     *                                                    *
C     ******************************************************
C
C
      DIMENSION SIGMA(4,2)
      INTEGER SHOCK,CNTACT,HALT
      DOUBLE PRECISION SIGMA
C
      PRINT * ,'CONDITION 7S HAS BEEN MET.  THIS MEANS THAT THE SHOCK '
      PRINT * ,'AND CONTACT SURFACE ARE LOCATED AT THE SAME X AT A    '
```

```
      PRINT * ,'TIME OTHER THAN ZERO.  THIS SUBROUTINE WOULD HAVE TO  '
      PRINT * ,'DETERMINE THE RESULT OF THE INTERSECTION BASED ON THE '
      PRINT * ,'CONDITIONS TO THE LEFT AND RIGHT.  ADDITIONAL LOGIC IS'
      PRINT * ,'REQUIRED TO PROCEED.'
      PRINT * ,'SHOCK =',SHOCK,'  CONTACT SURFACE =',CNTACT
      PRINT * ,'SIGMA(1,1) =',SIGMA(1,1),'  SIGMA(2,1) =',SIGMA(2,1)
      HALT = 1
      RETURN
      END
C
      SUBROUTINE COND8(SHOCK,CNTACT,HALT)
C
C     ****************************************************
C     *                                                  *
C     *              SUBROUTINE CONDITION 8              *
C     *                                                  *
C     ****************************************************
C
      INTEGER HALT,SHOCK,CNTACT
C
      PRINT * ,'CONDITION 8 COULD RESULT ONLY AFTER THE ORIGINAL SHOCK'
      PRINT * ,'HAS CROSSED THE CONTACT SURFACE, AND THE SUBSEQUENT   '
      PRINT * ,'CONDITIONS DETERMINED. ADDITIONAL LOGIC IS REQUIRED TO'
      PRINT * ,'CONTINUE.'
      PRINT * ,'SHOCK =',SHOCK,'  CONTACT SURFACE =',CNTACT
      HALT = 1
      RETURN
      END
C
      SUBROUTINE CORRCT(LNODE,RNODE,N,SIGMA,H,QQ,RR,S,G,G1,G2,I2,X2,W,
     C                  AR,DQ,VS,A,Q)
C
C     ****************************************************
C     *                                                  *
C     *        DISCONTINUITY CORRECTION SUBROUTINE       *
C     *                                                  *
C     ****************************************************
C
C    ------------ VARIABLE  DEFINITIONS ------------
C
C    ENTRPY - MODIFIED ENTROPY
C    SNDSPD - SONIC VELOCITY
C    UA - VELOCITY RELATIVE TO THE SHOCK, LEFT SIDE
C    UB - VELOCITY RELATIVE TO THE SHOCK, RIGHT SIDE
C    VLCTY - VELOCITY
C
      INTEGER LNODE,RNODE,N,I2,NODE,SHOCK,CNTACT,K
      DIMENSION LNODE(4),RNODE(4),SIGMA(4,2),QQ(N),RR(N),S(N),I2(4),
     C          X2(4),A(N),Q(N),XA(4),XB(4)
      DOUBLE PRECISION SIGMA,QQ,RR,S,A,Q,
     C                 H,G,G1,G2,X2,W,AR,DQ,VS,
     C                 RRA,RRB,QQA,QQB,AA,AB,QA,QB,SA,SB,
     C                 SA1,SA2,VLCTY,SNDSPD,ENTRPY,QQCALC,RRCALC,
     C                 XB,XA
C
C -----DEFINE STATEMENT FUNCTIONS-----
C
      QQCALC(VLCTY,SNDSPD,ENTRPY) = VLCTY + SNDSPD*ENTRPY
      RRCALC(VLCTY,SNDSPD,ENTRPY) = VLCTY - SNDSPD*ENTRPY
C
C -----SET INITIAL VALUES OF VARIABLES TO ZERO-----
C
      RRA = 0.D00
      RRB = 0.D00
      QQA = 0.D00
      QQB = 0.D00
        AA = 0.D00
        AB = 0.D00
        QA = 0.D00
```

```
          QB = 0.D00
          SA = 0.D00
          SB = 0.D00
          DO 15 K=1,4
               XA(K)=0.D0
               XB(K)=0.D0
   15  CONTINUE
C
C  -----DETERMINE IF ONLY ONE NODE NEEDS TO BE CORRECTED OR TWO-----
C  -----NODES.  ALSO, IF SHOCK AND CONTACT SURFACES ARE CLOSE-----
C  -----ENOUGH(WITH IN 2H) TO INTERACT-----
C
        IF((LNODE(2).EQ.100).OR.(LNODE(3).EQ.100)) THEN
                IF((LNODE(1).GE.(RNODE(1)-2)).AND.(RNODE(1).GT.0)) THEN
                        GO TO 20
                ELSE
                        GO TO 30
                END IF
        ELSE IF(RNODE(1).GT.0) THEN
                GO TO 20
        ELSE
                GO TO 10
        END IF
C
C -----BRANCH HERE IF ONLY ONE NODE TO BE CORRECTED (LNODE) AND,-----
C -----SHOCK/CONTACT SURFACE INTERACTION-----
C
   10 SHOCK = LNODE(2)
        CNTACT = LNODE(3)
C
C ---SHOCK ON LEFT, HEADED RIGHT, JUMPS OR NOT;CONTACT SURFACE ON-----
C ---RIGHT, HEADED LEFT, DOESN'T CROSS NODE ---
C
        IF((SHOCK.EQ.322.OR.SHOCK.EQ.321).AND.(CNTACT.EQ.232)) THEN
C
C -----DETERMINE NODE TO RIGHT OF SHOCK AND CONTACT SURFACE-----
C
                I2(1) = LNODE(1) + 1
                I2(2) = LNODE(1) + 1
                CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----EXTRAPOLATE TO RIGHT FACE OF CONTACT SURFACE-----
C
                IF (I2(2).EQ.N) THEN
                        CALL BBDRY(RR(N),QQ(N),S(N),RRB,QQB,SB,AB,QB)
                ELSE
                CALL EXTRAP(RR(I2(2)),RR(I2(2)+1),QQ(I2(2)),QQ(I2(2)+1),
     C                    S(I2(2)),S(I2(2)+1),XB(2),H,RRB,QQB,SB,AB,QB)
                END IF
C
C -----CALCULATE VARIABLE CHANGE ACROSS CONTACT SURFACE-----
C
                SA = S(I2(2)-1)
                CALL CSJUMP(AB,QB,SB,SA,G2,QA,AA)
C
C -----IF SHOCK INTERACTS CALCULATE CHANGE IN VARIABLES ACROSS SHOCK----
C
                IF(SHOCK.EQ.321) THEN
                        QB = QA
                        AB = AA
                        SB = SA
                        CALL SKJUMP(AB,QB,SB,AR,DQ,VS,G,G1,H,AA,QA,SA)
                        QQA = QQCALC(QA,AA,SA)
                        RRA = RRCALC(QA,AA,SA)
C
C -----INTERPOLATE TO NODE(I2(1)-1) AND ASSIGN CORRECTED VALUES-----
C
                        IF (I2(1).EQ.2) THEN
                                CALL BBDRY(RRA,QQA,SA,RR(I2(1)-1),QQ(I2(1)-1),
```

```
     C                         S(I2(1)-1),A(I2(1)-1),Q(I2(1)-1))
                      ELSE
                      CALL INTERP(RRA,RR(I2(1)-2),QQA,QQ(I2(1)-2),SA,
     C                     S(I2(1)-2),XA(1),(XA(1)+H),RR(I2(1)-1),
     C                     QQ(I2(1)-1),S(I2(1)-1),A(I2(1)-1),
     C                     Q(I2(1)-1))
                      END IF
                 ELSE
                      QQ(I2(2)-1) = QQCALC(QA,AA,SA)
                      RR(I2(2)-1) = RRCALC(QA,AA,SA)
                       S(I2(2)-1) = SA
                       A(I2(2)-1) = AA
                       Q(I2(2)-1) = QA
                 END IF
C
C ---SHOCK ON RIGHT, HEADED LEFT, CROSSES NODE OR NOT; CONTACT SURFACE-
C ---ON LEFT,HEADED RIGHT, DOES'NT CROSS NODE---
C
         ELSE IF((SHOCK.EQ.232.OR.SHOCK.EQ.231).AND.(CNTACT.EQ.322)) THEN
C
C -----DETERMINE NODE TO RIGHT TO SHOCK AND CONTACT SURFACE-----
C
                 I2(1) = LNODE(1)
                 I2(2) = LNODE(1)
                 CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----EXTRAPOLATE TO LEFT FACE OF CONTACT SURFACE-----
C
                 IF (I2(2).EQ.2) THEN
                      CALL BBDRY(RR(1),QQ(1),S(1),RRA,QQA,SA,AA,QA)
                 ELSE
                 CALL EXTRAP(RR(I2(2)-1),RR(I2(2)-2),QQ(I2(2)-1),QQ(I2(2)-2),
     C                 S(I2(2)-1),S(I2(2)-2),XA(2),H,RRA,QQA,SA,AA,QA)
                 END IF
C
C -----CALCULATE VARIABLE CHANGE ACROSS CONTACT SURFACE-----
C
                 SA = S(I2(2))
                 CALL CSJUMP(AA,QA,SA,SB,G2,QB,AB)
C
C -----IF SHOCK INTERACTS CALCULATE CHANGE IN VARIABLES ACROSS SHOCK----
C
                 IF(SHOCK.EQ.231) THEN
                      QA = QB
                      AA = AB
                      SA = SB
                      CALL SKJUMP(AA,QA,SA,AR,DQ,VS,G,G1,W,AB,QB,SB)
                      QQB = QQCALC(QB,AB,SB)
                      RRB = RRCALC(QB,AB,SB)
C
C -----INTERPOLATE TO NODE(I2(1)) AND ASSIGN CORRECTED VALUES-----
C
                      IF (I2(1).EQ.N) THEN
                          CALL BBDRY(RRB,QQB,SB,RR(I2(1)),QQ(I2(1)),S(I2(1)),
     C                       A(I2(1)),Q(I2(1)))
                      ELSE
                      CALL INTERP(RRB,RR(I2(1)+1),QQB,QQ(I2(1)+1),SB,
     C                     S(I2(1)+1),XB(1),(XB(1)+H),RR(I2(1)),
     C                     QQ(I2(1)),S(I2(1)),A(I2(1)),Q(I2(1)))
                      END IF
                 ELSE
                      QQ(I2(2)) = QQCALC(QB,AB,SB)
                      RR(I2(2)) = RRCALC(QB,AB,SB)
                        S(I2(2)) = SB
                        A(I2(2)) = AB
                        Q(I2(2)) = QB
                 END IF
C
C --- SHOCK ON RIGHT,HEADED RIGHT, DOESN'T CROSS OR ON LEFT,HEADED---
```

```
C --- RIGHT, DOES CROSS; AND CONTACT ON LEFT,HEADED RIGHT, CROSSES---
C --- OR NOT: OR SHOCK ON RIGHT, HEADED LEFT, DOESN'T CROSS NODE ---
C --- WITH CONTACT ON LEFT, HEADED RIGHT, AND CROSSED NODE ---
C
          ELSE IF(((SHOCK.EQ.222.OR.SHOCK.EQ.321).AND.(CNTACT.EQ.321.OR.
     C         CNTACT.EQ.322)).OR.(SHOCK.EQ.232.AND.CNTACT.EQ.321))
     C         THEN
C
C -----DETERMINE NODE TO RIGHT OF SHOCK AND CONTACT SURFACE-----
C
               I2(1) = LNODE(1) + 1
               I2(2) = LNODE(1) + 1
               CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----EXTRAPOLATE TO RIGHT FACE OF SHOCK-----
C
               IF (I2(1).EQ.N) THEN
                    CALL BBDRY(RR(N),QQ(N),S(N),RRB,QQB,SB,AB,QB)
               ELSE
               CALL EXTRAP(RR(I2(1)),RR(I2(1)+1),QQ(I2(1)),QQ(I2(1)+1),
     C                S(I2(1)),S(I2(1)+1),XB(1),H,RRB,QQB,SB,AB,QB)
               END IF
C
C -----CALCULATE CHANGE IN VARIABLES ACROSS SHOCK-----
C
               IF(SHOCK.EQ.232) THEN
                    AA = AB/AR
                    SA1 = (G1/G)*DLOG((2.D00*G*(W**2)-G+1.D00) / (G+1.D00))
                    SA2 = G1*DLOG(((G-1.D00)*(W**2)+2.D00) / ((G+1.D00)*
     C                     (W**2)))
                    SA = SB+SA1+SA2
                    UB = QB - VS
                    UA = UB -AA*DQ
                    QA = UA + VS
               ELSE
                    CALL SKJUMP(AB,QB,SB,AR,DQ,VS,G,G1,W,AA,QA,SA)
               END IF
C
C -----CALCULATE CHANGE IN VARIABLES ACROSS CONTACT SURFACE-----
C
               IF(CNTACT.EQ.322) THEN
                    QQ(I2(1)-1) = QQCALC(QA,AA,SA)
                    RR(I2(1)-1) = RRCALC(QA,AA,SA)
                     S(I2(1)-1) = SA
                     A(I2(1)-1) = AA
                     Q(I2(1)-1) = QA
               ELSE
                    QB = QA
                    AB = AA
                    SB = SA
                    SA = S(I2(2)-2)
                    CALL CSJUMP(AB,QB,SB,SA,G2,QA,AA)
                    QQA = QQCALC(QA,AA,SA)
                    RRA = RRCALC(QA,AA,SA)
C
C -----INTERPOLATE TO NODE(I2(2)-1) AND ASSIGN CORRECTED VALUES-----
C
                    IF (I2(2).EQ.2) THEN
                         CALL BBDRY(RRA,QQA,SA,RR(I2(2)-1),QQ(I2(2)-1),
     C                     S(I2(2)-1),A(I2(2)-1),Q(I2(2)-1))
                    ELSE
                    CALL INTERP(RRA,RR(I2(2)-2),QQA,QQ(I2(2)-2),SA,
     C                   S(I2(2)-2),XA(2),(XA(2)+H),RR(I2(2)-1),
     C                   QQ(I2(2)-1),S(I2(2)-1),A(I2(2)-1),
     C                   Q(I2(2)-1))
                    END IF
               END IF
C
C ---SHOCK ON LEFT, HEADED LEFT, DOESN'T CROSS NODE, OR ON RIGHT,---
```

```
C ---HEADED LEFT, CROSSED NODE; AND CONTACT ON RIGHT, HEADED LEFT,---
C ---CROSSES NODE OR NOT: OR SHOCK ON LEFT, HEADED RIGHT, DOESN'T---
C ---CROSS NODE, AND CONTACT ON RIGHT, HEADED LEFT, CROSSED NODE---
C
        ELSE IF((SHOCK.EQ.332.OR.SHOCK.EQ.231).AND.(CNTACT.EQ.232.OR.
     C          CNTACT.EQ.231)).OR.(SHOCK.EQ.322.AND.CNTACT.EQ.231))
     C       THEN
C
C -----DETERMINE NODE TO RIGHT OF SHOCK AND CONTACT SURFACE-----
C
                I2(1) = LNODE(1)
                I2(2) = LNODE(1)
                CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----EXTRAPOLATE TO LEFT FACE OF SHOCK-----
C
                IF (I2(1).EQ.2) THEN
                        CALL BBDRY(RR(1),QQ(1),S(1),RRA,QQA,SA,AA,QA)
                ELSE
                CALL EXTRAP(RR(I2(1)-1),RR(I2(1)-2),QQ(I2(1)-1),QQ(I2(1)-2),
     C                   S(I2(1)-1),S(I2(1)-2),XA(1),H,RRA,QQA,SA,AA,QA)
                END IF
C
C -----CALCULATE CHANGE IN VARIABLES ACROSS SHOCK-----
C
                IF(SHOCK.EQ.322) THEN
                        AB = AA/AR
                        SA1 = (G1/G)*DLOG((2.D00*G*(W**2)-G+1.D00) / (G+1.D00))
                        SA2 = G1*DLOG(((G-1.D00)*(W**2)+2.D00) / ((G+1.D00)*
     C                        (W**2)))
                        SB = SA+SA1+SA2
                        UA = QA - VS
                        UB = UA - AB*DQ
                        QB = UB + VS
                ELSE
                        CALL SKJUMP(AA,QA,SA,AR,DQ,VS,G,G1,W,AB,QB,SB)
                END IF
C
C -----CALCULATE CHANGE IN VARIABLES ACROSS CONTACT SURFACE-----
C
                IF(CNTACT.EQ.232) THEN
                        QQ(I2(1)) = QQCALC(QB,AB,SB)
                        RR(I2(1)) = RRCALC(QB,AB,SB)
                         S(I2(1)) = SB
                         A(I2(1)) = AB
                         Q(I2(1)) = QB
                ELSE
                        QA = QB
                        AA = AB
                        SA = SB
                        SB = S(I2(2)+1)
                        CALL CSJUMP(AA,QA,SA,SB,G2,QB,AB)
                        QQB = QQCALC(QB,AB,SB)
                        RRB = RRCALC(QB,AB,SB)
C
C -----INTERPOLATE TO NODE(I2(2)) AND ASSIGN CORRECTED VALUES-----
C
                        IF (I2(2).EQ.N) THEN
                            CALL BBDRY(RRB,QQB,SB,RR(I2(2)),QQ(I2(2)),S(I2(2)),
     C                          A(I2(2)),Q(I2(2)))
                        ELSE
                        CALL INTERP(RRB,RR(I2(2)+1),QQB,QQ(I2(2)+1),SB,
     C                       S(I2(2)+1),XB(2),(XB(2)+H),RR(I2(2)),
     C                       QQ(I2(2)),S(I2(2)),A(I2(2)),Q(I2(2)))
                        END IF
                END IF
        END IF
        GO TO 40
C
```

```
C -----BRANCH HERE IF LNODE AND RNODE ARE CLOSE ENOUGH FOR SHOCK-----
C -----AND CONTACT SURFACE INTERACTION-----
C
C LEFT NODE:
C ---SHOCK ON RIGHT, HEADED RIGHT, JUMPS NODE WITH CONTACT ON LEFT---
C ---HEADED RIGHT, JUMPS NODE OR NOT; OR NO SHOCK WITH CONTACT ON ---
C ---LEFT, HEADED RIGHT, JUMPS NODE---
C RIGHT NODE:
C ---SHOCK ON LEFT, HEADED RIGHT, JUMPS NODE WITH NO CONTACT SURFACE---
C
   20 IF(((LNODE(2).EQ.221).AND.(LNODE(3).EQ.321.OR.LNODE(3).EQ.322))
     C    .OR.(LNODE(2).EQ.100.AND.LNODE(3).EQ.321)) THEN
               IF((RNODE(2).EQ.321).AND.(RNODE(3).EQ.100)) THEN
C
C -----DETERMINE NODE TO RIGHT OF SHOCK AND CONTACT SURFACE-----
C
                       I2(1) = RNODE(1) + 1
                       IF(LNODE(2).EQ.322) THEN
                               I2(2) = LNODE(1)
                       ELSE
                               I2(2) = LNODE(1) + 1
                       END IF
                       CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----EXTRAPOLATE TO RIGHT FACE OF SHOCK-----
C
               IF (I2(1).EQ.N) THEN
                       CALL BBDRY(RR(N),QQ(N),S(N),RRB,QQB,SB,AB,QB)
               ELSE
               CALL EXTRAP(RR(I2(1)),RR(I2(1)+1),QQ(I2(1)),QQ(I2(1)+1),
     C                 S(I2(1)),S(I2(1)+1),XB(1),H,RRB,QQB,SB,AB,QB)
               END IF
C
C -----CALCULATE VARIABLE CHANGE ACROSS SHOCK-----
C
               CALL SKJUMP(AB,QB,SB,AR,DQ,VS,G,G1,W,AA,QA,SA)
C
C -----DETERMINE CORRECTED VALUES AT NODE(I2(1)-1) AND ASSIGN THEM-----
C
               IF(LNODE(2).EQ.100) THEN
                       QQA = QQCALC(QA,AA,SA)
                       RRA = RRCALC(QA,AA,SA)
C
C -----INTERPOLATE TO NODE(I2(1)-1) AND ASSIGN CORRECTED VALUES-----
C
                       IF (I2(1).EQ.2) THEN
                               CALL BBDRY(RRA,QQA,SA,RR(I2(1)-1),QQ(I2(1)-1),
     C                           S(I2(1)-1),A(I2(1)-1),Q(I2(1)-1))
                       ELSE
                       CALL INTERP(RRA,RR(I2(1)-2),QQA,QQ(I2(1)-2),SA,
     C                         S(I2(1)-2),XA(1),(XA(1)+H),RR(I2(1)-1),
     C                         QQ(I2(1)-1),S(I2(1)-1),A(I2(1)-1),
     C                         Q(I2(1)-1))
                       END IF
C
C -----EXTRAPOLATE TO RIGHT FACE OF CONTACT SURFACE-----
C
                       IF (I2(2).EQ.N) THEN
                               CALL BBDRY(RR(N),QQ(N),S(N),RRB,QQB,SB,AB,QB)
                       ELSE
                       CALL EXTRAP(RR(I2(2)),RR(I2(2)+1),QQ(I2(2)),QQ(I2(2)+1),
     C                        S(I2(2)),S(I2(2)+1),XB(2),H,RRB,QQB,SB,AB,QB)
                       END IF
               ELSE
                       QQ(I2(1)-1) = QQCALC(QA,AA,SA)
                       RR(I2(1)-1) = RRCALC(QA,AA,SA)
                        S(I2(1)-1) = SA
                        A(I2(1)-1) = AA
                        Q(I2(1)-1) = QA
```

```
                    IF(LNODE(3).EQ.322) THEN
                            QQ(I2(2)) = QQ(I2(1)-1)
                            RR(I2(2)) = RR(I2(1)-1)
                             S(I2(2)) = S(I2(1)-1)
                             A(I2(2)) = A(I2(1)-1)
                             Q(I2(2)) = Q(I2(1)-1)
                             GO TO 40
                    ELSE
                            QB = QA
                            AB = AA
                            SB = SA
                    END IF
            END IF
                    SA = S(I2(2)-2)
                    CALL CSJUMP(AB,QB,SB,SA,G2,QA,AA)
                    QQA = QQCALC(QA,AA,SA)
                    RRA = RRCALC(QA,AA,SA)
C
C -----INTERPOLATE TO NODE(I2(2)-1) AND ASSIGN CORRECTED VALUES-----
C
                    IF (I2(2).EQ.2) THEN
                            CALL BBDRY(RRA,QQA,SA,RR(I2(2)-1),QQ(I2(2)-1),
     C                        S(I2(2)-1),A(I2(2)-1),Q(I2(2)-1))
                    ELSE
                    CALL INTERP(RRA,RR(I2(2)-2),QQA,QQ(I2(2)-2),SA,
     C                      S(I2(2)-2),XA(2),(XA(2)+H),RR(I2(2)-1),
     C                      QQ(I2(2)-1),S(I2(2)-1),A(I2(2)-1),
     C                      Q(I2(2)-1))
                    END IF
            END IF
C
C LEFT NODE:
C ---SHOCK ON LEFT, HEADED RIGHT, JUMPS NODE WITH NO CONTACT---
C RIGHT NODE:
C ---NO SHOCK, WITH CONTACT ON RIGHT, HEADED LEFT,JUMPS NODE---
C
        ELSE IF((LNODE(2).EQ.321).AND.(LNODE(3).EQ.100)) THEN
              IF((RNODE(2).EQ.100).AND.(RNODE(3).EQ.231)) THEN
C
C -----DETERMINE NODE TO RIGHT OF SHOCK AND CONTACT SURFACE-----
C
                    I2(1) = LNODE(1) + 1
                    I2(2) = RNODE(1)
                    CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----CALCULATE JUMP THROUGH CONTACT SURFACE THEN SHOCK-----
C
                    IF(I2(1).EQ.(I2(2)-1)) THEN
                            AA = A(I2(1))
                            QA = Q(I2(1))
                            SA = S(I2(1))
                            SB = S(I2(2)+1)
                            CALL CSJUMP(AA,QA,SA,SB,G2,QB,AB)
                            QQB = QQCALC(QB,AB,SB)
                            RRB = RRCALC(QB,AB,SB)
C
C -----INTERPOLATE TO NODE(I2(2)) AND ASSIGN CORRECTED VALUES-----
C
                            IF (I2(2).EQ.N) THEN
                          CALL BBDRY(RRB,QQB,SB,RR(I2(2)),QQ(I2(2)),S(I2(2)),
     C                        A(I2(2)),Q(I2(2)))
                            ELSE
                            CALL INTERP(RRB,RR(I2(2)+1),QQB,QQ(I2(2)+1),SB,
     C                         S(I2(2)+1),XB(2),(XB(2)+H),RR(I2(2)),
     C                         QQ(I2(2)),S(I2(2)),A(I2(2)),Q(I2(2)))
                            END IF
                            AB = A(I2(1))
                            QB = Q(I2(1))
                            SB = S(I2(1))
```

```
                                CALL SKJUMP(AB,QB,SB,AR,DQ,VS,G,G1,W,AA,QA,SA)
                                QQA = QQCALC(QA,AA,SA)
                                RRA = RRCALC(QA,AA,SA)
C
C -----INTERPOLATE TO NODE(I2(1)-1) AND ASSIGN CORRECTED VALUES-----
C
                                IF (I2(1).EQ.2) THEN
                                CALL BBDRY(RRA,QQA,SA,RR(I2(1)-1),QQ(I2(1)-1),
     C                            S(I2(1)-1),A(I2(1)-1),Q(I2(1)-1))
                                ELSE
                                CALL INTERP(RRA,RR(I2(1)-2),QQA,QQ(I2(1)-2),SA,
     C                            S(I2(1)-2),XA(1),(XA(1)+H),RR(I2(1)-1),
     C                            QQ(I2(1)-1),S(I2(1)-1),A(I2(1)-1),
     C                            Q(I2(1)-1))
                                END IF
                         ELSE
                                RR(I2(1)-1) = RR(I2(1)-2)
                                QQ(I2(1)-1) = QQ(I2(1)-2)
                                 S(I2(1)-1) = S(I2(1)-2)
                                 A(I2(1)-1) = A(I2(1)-2)
                                 Q(I2(1)-1) = Q(I2(1)-2)
                                RR(I2(2)) = RR(I2(2)+1)
                                QQ(I2(2)) = QQ(I2(2)+1)
                                 S(I2(2)) = S(I2(2)+1)
                                 A(I2(2)) = A(I2(2)+1)
                                 Q(I2(2)) = Q(I2(2)+1)
                         END IF
                  END IF
C
C RIGHT NODE:
C ---SHOCK ON RIGHT, HEADED LEFT, JUMPS NODE WITH NO CONTACT---
C LEFT NODE:
C ---NO SHOCK, WITH CONTACT ON LEFT, HEADED RIGHT, JUMPS NODE---
C
            ELSE IF((RNODE(2).EQ.231).AND.(RNODE(3).EQ.100)) THEN
                  IF((LNODE(2).EQ.100).AND.(LNODE(3).EQ.321)) THEN
C
C -----DETERMINE NODE TO RIGHT TO SHOCK AND CONTACT SURFACE-----
C
                         I2(1) = RNODE(1)
                         I2(2) = LNODE(1) + 1
                         CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----CALCULATE JUMP THROUGH CONTACT SURFACE THEN SHOCK-----
C
                         IF(I2(2).EQ.(I2(1)-1)) THEN
                                AB = A(I2(2))
                                QB = Q(I2(2))
                                SB = S(I2(2))
                                SA = S(I2(2)-2)
                                CALL CSJUMP(AB,QB,SB,SA,G2,QA,AA)
                                QQA = QQCALC(QA,AA,SA)
                                RRA = RRCALC(QA,AA,SA)
C
C -----INTERPOLATE TO NODE(I2(2)-1) AND ASSIGN CORRECTED VALUES-----
C
                         IF (I2(2).EQ.2) THEN
                                CALL BBDRY(RRA,QQA,SA,RR(I2(2)-1),QQ(I2(2)-1),
     C                            S(I2(2)-1),A(I2(2)-1),Q(I2(2)-1))
                         ELSE
                         CALL INTERP(RRA,RR(I2(2)-2),QQA,QQ(I2(2)-2),SA,
     C                          S(I2(2)-2),XA(2),(XA(2)+H),RR(I2(2)-1),
     C                          QQ(I2(2)-1),S(I2(2)-1),A(I2(2)-1),
     C                          Q(I2(2)-1))
                         END IF
                                AA = A(I2(2))
                                QA = Q(I2(2))
                                SA = S(I2(2))
                                CALL SKJUMP(AA,QA,SA,AR,DQ,VS,G,G1,W,AB,QB,SB)
```

```
                                 QQB = QQCALC(QB,AB,SB)
                                 RRB = RRCALC(QB,AB,SB)
C
C -----INTERPOLATE TO NODE(I2(1)) AND ASSIGN CORRECTED VALUES-----
C
                                 IF (I2(1).EQ.N) THEN
                                CALL BBDRY(RRB,QQB,SB,RR(I2(1)),QQ(I2(1)),S(I2(1)),
     C                             A(I2(1)),Q(I2(1)))
                                 ELSE
                                 CALL INTERP(RRB,RR(I2(1)+1),QQB,QQ(I2(1)+1),SB,
     C                              S(I2(1)+1),XB(1),(XB(1)+H),RR(I2(1)),
     C                              QQ(I2(1)),S(I2(1)),A(I2(1)),Q(I2(1)))
                                 END IF
                          ELSE
                                 RR(I2(2)-1) = RR(I2(2)-2)
                                 QQ(I2(2)-1) = QQ(I2(2)-2)
                                  S(I2(2)-1) = S(I2(2)-2)
                                  A(I2(2)-1) = A(I2(2)-2)
                                  Q(I2(2)-1) = Q(I2(2)-2)
                                 RR(I2(1)) = RR(I2(1)+1)
                                 QQ(I2(1)) = QQ(I2(1)+1)
                                  S(I2(1)) = S(I2(1)+1)
                                  A(I2(1)) = A(I2(1)+1)
                                  Q(I2(1)) = Q(I2(1)+1)
                          END IF
                   END IF
C
C RIGHT NODE:
C ---SHOCK ON LEFT, HEADED LEFT, JUMPS NODE WITH CONTACT ON RIGHT---
C ---HEADED LEFT, DOES NOT CROSS NODE OR NO SHOCK WITH CONTACT ON---
C ---RIGHT, HEADED LEFT, CROSSED NODE---
C LEFT NODE:
C ---SHOCK ON RIGHT, HEADED LEFT, JUMPS NODE WITH NO CONTACT---
C
            ELSE IF(((RNODE(2).EQ.331).AND.(RNODE(3).EQ.231.OR.RNODE(3).EQ.
     C              232)).OR.(RNODE(2).EQ.100.AND.RNODE(3).EQ.231)) THEN
                   IF((LNODE(2).EQ.231).AND.(LNODE(3).EQ.100)) THEN
C
C -----DETERMINE NODE TO RIGHT OF SHOCK AND CONTACT SURFACE-----
C
                          I2(1) = LNODE(1)
                          IF(RNODE(3).EQ.232) THEN
                                 I2(2) = RNODE(1) + 1
                          ELSE
                                 I2(2) = RNODE(1)
                          END IF
                          CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----EXTRAPOLATE TO LEFT FACE OF SHOCK-----
C
                   IF (I2(1).EQ.2) THEN
                          CALL BBDRY(RR(1),QQ(1),S(1),RRA,QQA,SA,AA,QA)
                   ELSE
                   CALL EXTRAP(RR(I2(1)-1),RR(I2(1)-2),QQ(I2(1)-1),QQ(I2(1)-2),
     C                   S(I2(1)-1),S(I2(1)-2),XA(1),H,RRA,QQA,SA,AA,QA)
                   END IF
C
C -----CALCULATE VARIABLE CHANGE ACROSS SHOCK-----
C
                   CALL SKJUMP(AA,QA,SA,AR,DQ,VS,G,G1,W,AB,QB,SB)
C
C -----DETERMINE CORRECTED VALUES AT NODE(I2(1)) AND ASSIGN THEM-----
C
                   IF(RNODE(2).EQ.100) THEN
                          QQB = QQCALC(QB,AB,SB)
                          RRB = RRCALC(QB,AB,SB)
C
C -----INTERPOLATE TO NODE(I2(1)) AND ASSIGN CORRECTED VALUES-----
C
```

```
                           IF (I2(1).EQ.N) THEN
                          CALL BBDRY(RRB,GQB,SB,RR(I2(1)),QQ(I2(1)),S(I2(1)),
     C                       A(I2(1)),Q(I2(1)))
                           ELSE
                           CALL INTERP(RRB,RR(I2(1)+1),QQB,QQ(I2(1)+1),SB,
     C                         S(I2(1)+1),XB(1),(XB(1)+H),RR(I2(1)),
     C                         QQ(I2(1)),S(I2(1)),A(I2(1)),Q(I2(1)))
                           END IF
C
C -----EXTRAPOLATE TO LEFT FACE OF CONTACT SURFACE-----
C
                IF (I2(2).EQ.2) THEN
                      CALL BBDRY(RR(1),QQ(1),S(1),RRA,QQA,SA,AA,QA)
                ELSE
                CALL EXTRAP(RR(I2(2)-1),RR(I2(2)-2),QQ(I2(2)-1),QQ(I2(2)-2),
     C                 S(I2(2)-1),S(I2(2)-2),XA(2),H,RRA,QQA,SA,AA,QA)
                END IF
                ELSE
                      QQ(I2(1)) = QQCALC(QB,AB,SB)
                      RR(I2(1)) = RRCALC(QB,AB,SB)
                       S(I2(1)) = SB
                       A(I2(1)) = AB
                       Q(I2(1)) = QB
                      IF(RNODE(3).EQ.232) THEN
                           RR(I2(1)+1) = RR(I2(1))
                           QQ(I2(1)+1) = QQ(I2(1))
                            S(I2(1)+1) = S(I2(1))
                            A(I2(1)+1) = A(I2(1))
                            Q(I2(1)+1) = Q(I2(1))
                            GO TO 40
                      ELSE
                           QA = QB
                           AA = AB
                           SA = SB
                      END IF
                END IF
                      SB = S(I2(2)+1)
C
C -----CALCULATE VARIABLE CHANGE ACROSS CONTACT SURFACE-----
C
                      CALL CSJUMP(AA,QA,SA,SB,G2,QB,AB)
                      QQB = QQCALC(QB,AB,SB)
                      RRB = RRCALC(QB,AB,SB)
C
C -----INTERPOLATE TO NODE(I2(2)) AND ASSIGN CORRECTED VALUES-----
C
                           IF (I2(2).EQ.N) THEN
                          CALL BBDRY(RRB,QQB,SB,RR(I2(2)),QQ(I2(2)),S(I2(2)),
     C                       A(I2(2)),Q(I2(2)))
                           ELSE
                           CALL INTERP(RRB,RR(I2(2)+1),QQB,QQ(I2(2)+1),SB,
     C                         S(I2(2)+1),XB(2),(XB(2)+H),RR(I2(2)),
     C                         QQ(I2(2)),S(I2(2)),A(I2(2)),Q(I2(2)))
                           END IF
                END IF
           END IF
           GO TO 40
C
C --- BRANCH HERE IF THERE ARE ONE OR TWO NODES TO CORRECT, BUT THEY ---
C --- ARE SEPERATED BY MORE THAN 2H.  CHECK FOR SHOCK/CONTACT SURFACE---
C --- INTERACTION, AND CORRECT APPROPRIATELY ---
C
   30 NODE = LNODE(1)
           SHOCK = LNODE(2)
           CNTACT = LNODE(3)
C
C --- NO SHOCK, CONTACT SURFACE ON LEFT, HEADED RIGHT, JUMPS NODE ---
C
   35 IF((SHOCK.EQ.100).AND.(CNTACT.EQ.321)) THEN
```

```
            I2(2) = NODE + 1
            CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----CHECK FOR A SHOCK INTERACTING WITH CONTACT SURFACE AT NODE-----
C
            IF((SIGMA(1,2).LT.(X2(2)+H)).AND.
     C      (SIGMA(1,2).GT.(X2(2))))THEN
                  AB = A(I2(2))
                  QB = Q(I2(2))
                  SB = S(I2(2))
            ELSE
C
C -----EXTRAPOLATE TO RIGHT FACE OF CONTACT SURFACE-----
C
                  IF (I2(2).EQ.N) THEN
                        CALL BBDRY(RR(N),QQ(N),S(N),RRB,QQB,SB,AB,QB)
                  ELSE
                  CALL EXTRAP(RR(I2(2)),RR(I2(2)+1),QQ(I2(2)),QQ(I2(2)+1),
     C                 S(I2(2)),S(I2(2)+1),XB(2),H,RRB,QQB,SB,AB,QB)
                  END IF
            END IF
            SA = S(I2(2)-2)
C
C --- CALCULATE VARIABLE CHANGE OVER CONTACT SURFACE ---
C
            CALL CSJUMP(AB,QB,SB,SA,G2,QA,AA)
            QQA = QQCALC(QA,AA,SA)
            RRA = RRCALC(QA,AA,SA)
C
C -----INTERPOLATE TO NODE(I2(2)-1) AND ASSIGN CORRECTED VALUES-----
C
            IF (I2(2).EQ.2) THEN
                  CALL BBDRY(RRA,QQA,SA,RR(I2(2)-1),QQ(I2(2)-1),
     C                S(I2(2)-1),A(I2(2)-1),Q(I2(2)-1))
            ELSE
            CALL INTERP(RRA,RR(I2(2)-2),QQA,QQ(I2(2)-2),SA,
     C              S(I2(2)-2),XA(2),(XA(2)+H),RR(I2(2)-1),
     C              QQ(I2(2)-1),S(I2(2)-1),A(I2(2)-1),
     C              Q(I2(2)-1))
            END IF
C
C --- NO SHOCK, CONTACT SURFACE ON LEFT, HEADED LEFT, JUMPS NODE ---
C
        ELSE IF((SHOCK.EQ.100).AND.(CNTACT.EQ.231)) THEN
            I2(2) = NODE
            CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----CHECK FOR A SHOCK INTERACTING WITH CONTACT SURFACE AT NODE-----
C
            IF((SIGMA(1,2).GT.(X2(2)-(2.D00*H))).AND.(SIGMA(1,2).LT.
     C      (X2(2)-H))) THEN
                  AA = A(I2(2)-1)
                  QA = Q(I2(2)-1)
                  SA = S(I2(2)-1)
            ELSE
C
C -----EXTRAPOLATE TO LEFT FACE OF CONTACT SURFACE-----
C
            IF (I2(2).EQ.2) THEN
                  CALL BBDRY(RR(1),QQ(1),S(1),RRA,QQA,SA,AA,QA)
            ELSE
            CALL EXTRAP(RR(I2(2)-1),RR(I2(2)-2),QQ(I2(2)-1),QQ(I2(2)-2),
     C              S(I2(2)-1),S(I2(2)-2),XA(2),H,RRA,QQA,SA,AA,QA)
            END IF
            END IF
            SB = S(I2(2)+1)
            CALL CSJUMP(AA,QA,SA,SB,G2,QB,AB)
            QQB = QQCALC(QB,AB,SB)
            RRB = RRCALC(QB,AB,SB)
```

```
C
C -----INTERPOLATE TO NODE(I2(2)) AND ASSIGN CORRECTED VALUES-----
C
              IF (I2(2).EQ.N) THEN
                   CALL BBDRY(RRB,QQB,SB,RR(I2(2)),QQ(I2(2)),S(I2(2)),
      C                 A(I2(2)),Q(I2(2)))
              ELSE
                   CALL INTERP(RRB,RR(I2(2)+1),QQB,QQ(I2(2)+1),SB,
      C                  S(I2(2)+1),XB(2),(XB(2)+H),RR(I2(2)),
      C                  QQ(I2(2)),S(I2(2)),A(I2(2)),Q(I2(2)))
              END IF
C
C --- SHOCK ON LEFT, HEADED RIGHT, JUMPS NODE WITH NO CONTACT ---
C
              ELSE IF((SHOCK.EQ.321).AND.(CNTACT.EQ.100)) THEN
              I2(1) = NODE + 1
              CALL DELTAX(I2,SIGMA,XB,XA,H)
C
C -----CHECK FOR A CONTACT SURFACE INTERACTING WITH SHOCK AT NODE-----
C
              IF((SIGMA(2,2).LT.(X2(1)+H)).AND.
      C        (SIGMA(2,2).GT.(X2(1)))) THEN
                    AB = A(I2(1))
                    QB = Q(I2(1))
                    SB = S(I2(1))
              ELSE
C
C -----EXTRAPOLATE TO RIGHT FACE OF SHOCK-----
C
                    IF (I2(1).EQ.N) THEN
                          CALL BBDRY(RR(N),QQ(N),S(N),RRB,QQB,SB,AB,QB)
                    ELSE
                    CALL EXTRAP(RR(I2(1)),RR(I2(1)+1),QQ(I2(1)),QQ(I2(1)+1),
      C                  S(I2(1)),S(I2(1)+1),XB(1),H,RRB,QQB,SB,AB,QB)
                    END IF
              END IF
C
C --- CALCULATE VARIABLE CHANGE ACROSS SHOCK ---
C
              CALL SKJUMP(AB,QB,SB,AR,DQ,VS,G,G1,H,AA,QA,SA)
              QQA = QQCALC(QA,AA,SA)
              RRA = RRCALC(QA,AA,SA)
C
C -----INTERPOLATE TO NODE(I2(1)-1) AND ASSIGN CORRECTED VALUES-----
C
              IF (I2(1).EQ.2) THEN
                    CALL BBDRY(RRA,QQA,SA,RR(I2(1)-1),QQ(I2(1)-1),
      C                  S(I2(1)-1),A(I2(1)-1),Q(I2(1)-1))
              ELSE
              CALL INTERP(RRA,RR(I2(1)-2),QQA,QQ(I2(1)-2),SA,
      C                S(I2(1)-2),XA(1),(XA(1)+H),RR(I2(1)-1),
      C                QQ(I2(1)-1),S(I2(1)-1),A(I2(1)-1),
      C                Q(I2(1)-1))
              END IF
C
C --- SHOCK ON RIGHT, HEADED LEFT, JUMPS NODE WITH NO CONTACT SURFACE--
C
         ELSE IF((SHOCK.EQ.231).AND.(CNTACT.EQ.100)) THEN
              I2(1) = NODE
              CALL DELTAX(I2,SIGMA,XB,XA,H)
              IF((SIGMA(2,2).GT.(X2(1)-(2.D00*H))).AND.(SIGMA(2,2).LT.
      C       (X2(1)-H))) THEN
                    AA = A(I2(1)-1)
                    QA = Q(I2(1)-1)
                    SA = S(I2(1)-1)
              ELSE
C
C -----EXTRAPOLATE TO LEFT FACE OF CONTACT SURFACE-----
C
```

```
                IF (I2(1).EQ.2) THEN
                       CALL BBDRY(RR(1),QQ(1),S(1),RRA,QQA,SA,AA,QA)
                ELSE
                CALL EXTRAP(RR(I2(1)-1),RR(I2(1)-2),QQ(I2(1)-1),QQ(I2(1)-2),
     C                S(I2(1)-1),S(I2(1)-2),XA(1),H,RRA,QQA,SA,AA,QA)
                END IF
                END IF
C
C --- CALCULATE VARIABLE CHANGE OVER SHOCK ---
C
        IF(I2(1).EQ.N-1) THEN
        END IF
                CALL SKJUMP(AA,QA,SA,AR,DQ,VS,G,G1,H,AB,QB,SB)
C
C -----DETERMINE CORRECTED VALUES AT NODE(I2(1)) AND ASSIGN THEM-----
C
        IF(I2(1).EQ.N-1) THEN
        END IF
                QQB = QQCALC(QB,AB,SB)
                RRB = RRCALC(QB,AB,SB)
C
C -----INTERPOLATE TO NODE(I2(1)) AND ASSIGN CORRECTED VALUES-----
C
                IF (I2(1).EQ.N) THEN
                     CALL BBDRY(RRB,QQB,SB,RR(I2(1)),QQ(I2(1)),S(I2(1)),
     C                    A(I2(1)),Q(I2(1)))
                ELSE
                CALL INTERP(RRB,RR(I2(1)+1),QQB,QQ(I2(1)+1),SB,
     C                S(I2(1)+1),XB(1),(XB(1)+H),RR(I2(1)),
     C                QQ(I2(1)),S(I2(1)),A(I2(1)),Q(I2(1)))
                END IF
        END IF
C
C --- CHECK IF SECOND NODE NEEDS CORRECTING IF SO LOOP BACK AROUND ---
C
        IF(RNODE(1).EQ.0) GO TO 40
                NODE = RNODE(1)
                SHOCK = RNODE(2)
                CNTACT = RNODE(3)
                RNODE(1) = 0
                GO TO 35
   40 RETURN
        END
C
        SUBROUTINE DELTAX(I2,SIGMA,XB,XA,H)
C
C       ******************************************************
C       *                                                    *
C       *   LOCATE DISCONTINUITY WITHIN INTERVAL SUBROUTINE  *
C       *                                                    *
C       ******************************************************
C
        DIMENSION I2(4),SIGMA(4,2),XB(4),XA(4),X2(4)
        INTEGER I2
        DOUBLE PRECISION SIGMA,XB,XA,H,X2
        X2(3)=0.D00
        X2(4)=0.D00
C
C -----CALCULATE DISTANCE NODE TO RIGHT OF DISCONTINUITY IS FROM LEFT --
C -----BOUNDARY OF TUBE THEN DETERMINE DISTANCE FROM THIS NODE I2(*)----
C -----TO DISCONTINUITY AND DISTANCE FROM NODE TO LEFT OF DISCONTINUITY-
C -----TO THAT DISCONTINUITY-----
C
        X2(1) = DBLE(I2(1)-1) * H
        XB(1) = (X2(1)-SIGMA(1,2))
        XA(1) = (H-XB(1))
C
C -----CALCULATE SAME VALUES FOR CONTACT SURFACE-----
C
```

```
      X2(2) = DBLE(I2(2)-1) * H
      XB(2) = (X2(2)-SIGMA(2,2))
      XA(2) = (H-XB(2))
      RETURN
      END
C
      SUBROUTINE SKJUMP(ADN,QDN,SDN,ARATIO,DELTAQ,VSHOCK,G,G1,W,
     C                   AUP,QUP,SUP)
C
C     ****************************************************
C     *                                                  *
C     *            SHOCK JUMP SUBROUTINE                 *
C     *                                                  *
C     ****************************************************
C
C   ------------ VARIABLE  DEFINITIONS ------------
C
C   *DN - VARIABLE DOWNSTREAM OF SHOCK
C   ARATIO - SPEED OF SOUND RATIO
C   DELTAQ - VELOCITY JUMP ACROSS SHOCK
C   *UP - VARIABLE UPSTREAM OF SHOCK
C   VSHOCK - SHOCK VELOCITY
C
      DOUBLE PRECISION ADN,QDN,SDN,ARATIO,DELTAQ,VSHOCK,G,G1,W,
     C                 AUP,QUP,SUP,SA1,SA2,UDN,UUP
C
C -----CALCULATE THE SPEED OF SOUND CHANGE THRU SHOCK-----
C
      AUP = ADN*ARATIO
C
C -----CALCULATE THE ENTROPY CHANGE THRU SHOCK-----
C
      SA1 = (G1/G)*DLOG((2.D00*G*(W**2)-G+1.D00)/(G+1.D00))
      SA2 = G1*DLOG(((G-1.D00)*(W**2)+2)/((G+1.D00)*(W**2)))
      SUP = SDN - SA1 - SA2
C
C -----CALCULATE THE VELOCITY CHANGE ACROSS SHOCK-----
C
      UDN = QDN - VSHOCK
      UUP = UDN + ADN*DELTAQ
      QUP = UUP + VSHOCK
      RETURN
      END
C
      SUBROUTINE CSJUMP(ADN,QDN,SDN,SUP,G2,QUP,AUP)
C
C     ****************************************************
C     *                                                  *
C     *       CONTACT SURFACE JUMP SUBROUTINE            *
C     *                                                  *
C     ****************************************************
C
      DOUBLE PRECISION QUP,QDN,AUP,ADN,SUP,SDN,G2
C
C -----CALCULATE THE SPEED OF SOUND CHANGE ACROSS THE CONTACT SURFACE---
C -----NOTE: THE VELOCITY IS CONTINUOUS ACROSS A C.S.-----
C
      QUP = QDN
      AUP = ADN * DEXP((SDN-SUP)/G2)
      RETURN
      END
C
      SUBROUTINE EXTRAP(RRL,RRR,QQL,QQR,SL,SR,DX,DH,RRINT,QQINT,SINT,
     C                   AINT,QINT)
C     ****************************************************
C     *                                                  *
C     *            EXTRAPOLATION SUBROUTINE              *
C     *                                                  *
C     ****************************************************
```

```
C
        DOUBLE PRECISION RRL,RRR,QQL,QQR,SL,SR,DX,DH,RRINT,QQINT,SINT,
      C                   AINT,QINT
C
C -----REIMAN VARIABLES ARE EXTRAPOLATED, AND THE CORRESPONDING-----
C -----VALUES FOR VELOCITY AND SPEED OF SOUND ARE CALCULATED-----
C
        RRINT = RRL - (DX * (RRR-RRL)/DH)
        QQINT = QQL - (DX * (QQR-QQL)/DH)
          SINT = SL - (DX * (SR-SL)/DH)
          AINT = (QQINT-RRINT)/(2.D00*SINT)
          QINT = (QQINT+RRINT)/(2.D00)
        RETURN
        END
C
        SUBROUTINE INTERP(RRL,RRR,QQL,QQR,SL,SR,DX,DH,RRINT,QQINT,SINT,
      C                    AINT,QINT)
C       **************************************************************
C       *                                                            *
C       *              INTERPOLATION SUBROUTINE                      *
C       *                                                            *
C       **************************************************************
C
        DOUBLE PRECISION RRL,RRR,QQL,QQR,SL,SR,DX,DH,RRINT,QQINT,SINT,
      C                   AINT,QINT
C
C -----REIMAN VARIABLES ARE INTERPOLATED, AND THEIR CORRESPONDING-----
C -----VALUES OF VELOCITY AND SPEED OF SOUND ARE CALCULATED-----
C
        RRINT = RRL - (DX * (RRL-RRR)/DH)
        QQINT = QQL - (DX * (QQL-QQR)/DH)
          SINT = SL - (DX * (SL-SR)/DH)
          AINT = (QQINT-RRINT)/(2.D00*SINT)
          QINT = (QQINT+RRINT)/2.D00
        RETURN
        END
C
        SUBROUTINE DBURST(N,H,QQ,RR,S,G,G1,G2,DELT,I2,X2,W,AR,DQ,VS,
      #LWPRES,SIGMA,A,Q)
C
C     ******************************************************
C     *                                                    *
C     *      DIAPHRAM BURSTING SUBROUTINE                  *
C     *                                                    *
C     ******************************************************
C
        INTEGER N,I,Y,I2,L,SHKDIR,CSDIR,LWPRES
        DIMENSION X2(4),RR(N),QQ(N),S(N),I2(4),SIGMA(4,2),A(N),Q(N)
        DOUBLE PRECISION X2,X,H,AB,SA,SB,AA,QA,QB,QQA,QQB,RRA,RRB,SIGMA,
      C                  RR,QQ,S,TS,W,DQ,AR,PR,G,G1,G2,VS,DELT,CSRMN,A,
      C                  Q,MREIMN,DREMN,EREIMN,WW,SA1,SA2,SAP,SBP,XB,XA
C
C     ++++++++  LOCATING THE NODE TO THE RIGHT +++++++++++
C
        DO 10 L=1,4
             SIGMA(L,1)=SIGMA(L,2)
             Y=0
             X=H
             I=2
   11      IF (.NOT.(Y.EQ.0)) GOTO 10
                IF (SIGMA(L,1).LT.X) THEN
                       X2(L)=X
                       I2(L)=I
                       Y=1
                END IF
             X=X+H
             I=I+1
             GOTO 11
   10   CONTINUE
```

```
C
C     +++++++++++++  CORRECT FOR DIAPHRAM BURSTING +++++++++++++
C
C
C -----AT TIME ZERO DETERMINE CORRECT SHOCK DIRECTION-----
C -----SHKDIR = 3 IS A SHOCK HEADED LEFT, AND SHKDIR = 2 IS SHOCK-----
C -----TRAVELING RIGHT-----
C
                  IF(LWPRES.EQ.3) THEN
                         SHKDIR = 3
                         X2(1) = X2(1) - H
                         I2(1) = I2(1) - 1
                         X2(2) = X2(2) - H
                         I2(2) = I2(2) - 1
                         GO TO 20
                  ELSE
                         SHKDIR = 2
                  END IF
C
   20 RRA=RR(I2(1)-1)
        RRB=RR(I2(1))
        QQA=QQ(I2(1)-1)
        QQB=QQ(I2(1))
        SA=S(I2(1)-1)
        SB=S(I2(1))
   21 AB=(QQB-RRB)/(2.D00*SB)
        AA=(QQA-RRA)/(2.D00*SA)
        IF(SHKDIR.EQ.3) THEN
                MREIMN = (RRB-RRA)/AA
                DREMN = DABS(MREIMN)
        ELSE
                MREIMN = (QQA-QQB)/AB
                DREMN = MREIMN
        END IF
C
C -----ITERATE FOR PROPER VALUE OF W USING THE QUADRATIC FIT OF THE-----
C -----REIMAN VARIABLE CHANGE WITH V CURVE.  NOTE LEFT MOVING SHOCKS----
C -----ARE USED IN THESE EQUATIONS SINCE RRB-RRA/AA=-(QQA-QQB/AB)-----
C
  100 WW=(3.0396408D01-((DREMN+2.7574D00)/0.286337D00))
        W=5.513294D00-DSQRT(WW)
        DQ=2.D00*(W*W-1.D00)/(W*(G+1.D00))
        AR=DSQRT(2.D00*(G-1.D00)*(1.D00+((G-1.D00)*W*W/2.D00))*
      C        (G*G2*W*W-1.D00))/((G+1.D00)*W)
        PR=(2.D00*G/(G+1.D00))*W*W-((G-1.D00)/(G+1.D00))
        DR=((G-1.D00)*W*W+2.D00)/((G+1.D00)*W*W)
        EREIMN=DQ+(AR-1.D00)*G2-(AR*G1/G)*DLOG(PR*(DR**G))
        IF (DABS(EREIMN-DABS(MREIMN)).LT.0.1D-5) GO TO 110
        DREMN = (DABS(MREIMN) - EREIMN) + DREMN
        GOTO 100
C
C -----DETERMINE S BEHIND SHOCK AND THEN CALCULATE THE RIEMANN-----
C -----VARIABLE CHANGE ACROSS THE CONTACT SURFACE-----
C
  110      SA1 = (G1/G)*DLOG((2.D00*G*(W**2)-G+1.D00)/(G+1.D00))
                SA2 = G1*DLOG(((G-1.D00)*(W**2)+2)/((G+1.D00)*(W**2)))
                IF(SHKDIR.EQ.2) THEN
                        SAP = SB - SA1 - SA2
                        SBP = SAP
                ELSE
                        SBP = SA - SA1 - SA2
                        SAP = SBP
                END IF
  103      IF(SHKDIR.EQ.2) THEN
                        CSRMN =((DEXP((SBP-SA)/G2))*(SA)-(SBP))*AR
                ELSE
                        CSRMN =((DEXP((SAP-SB)/G2))*(SB)-(SAP))*AR
                END IF
                IF(SHKDIR.EQ.3) THEN
```

```
                 MREIMN =((RRB-RRA)/AA)+CSRMN
                 DREMN = DABS(MREIMN)
          ELSE
                 MREIMN =((QQA-QQB)/AB)-CSRMN
                 DREMN = MREIMN
          END IF
  101     WW=(3.0396408D01-((DREMN+2.7574D00)/0.286337D00))
          W=5.513294D00-DSQRT(WW)
          DQ=2.D00*(W*W-1.D00)/(W*(G+1.D00))
          AR=DSQRT(2.D00*(G-1.D00)*(1.D00+((G-1.D00)*W*W/2.D00))*
     C        (G*G2*W*W-1.D00))/((G+1.D00)*W)
          PR=(2.D00*G/(G+1.D00))*W*W-((G-1.D00)/(G+1.D00))
          DR=((G-1.D00)*W*W+2.D00)/((G+1.D00)*W*W)
          EREIMN=DQ+(AR-1.D00)*G2-(AR*G1/G)*DLOG(PR*(DR**G))
          IF (DABS(EREIMN-DABS(MREIMN)).LT.0.1D-5) GO TO 102
          DREMN = (DABS(MREIMN) - EREIMN) + DREMN
          GOTO 101
  102     SA1 = (G1/G)*DLOG((2.D00*G*(W**2)-G+1.D00)/(G+1.D00))
          SA2 = G1*DLOG(((G-1.D00)*(W**2)+2)/((G+1.D00)*(W**2)))
          IF(SHKDIR.EQ.2) THEN
                 SAP = SB - SA1 - SA2
          ELSE
                 SBP = SA - SA1 - SA2
          END IF
          IF (DABS(SAP-SBP).LT.0.1D-5) GO TO 105
          IF(SHKDIR.EQ.2) THEN
               SBP = (SAP-SBP) + SBP
          ELSE
               SAP = (SBP-SAP) + SAP
          END IF
          GO TO 103
C
C ----- CALCULATE SHOCK VELOCITY -----
C
  105 IF(SHKDIR.EQ.3) THEN
          DQ = (-1.D00)*DQ
          VS = ((RRA+QQA)*0.5D00) - (W*AA)
      ELSE
          VS=(QQB+RRB)*0.5D00+W*AB
      END IF
      IF(SHKDIR.EQ.2) THEN
C
C -----EXTRAPOLATE TO RIGHT FACE OF SHOCK-----
C
          CALL EXTRAP(RR(I2(1)),RR(I2(1)+1),QQ(I2(1)),QQ(I2(1)+1),
     C             S(I2(1)),S(I2(1)+1),H,H,RRB,QQB,SB,AB,QB)
C
C -----CALCULATE CHANGE IN VARIABLES ACROSS SHOCK-----
C
                 CALL SKJUMP(AB,QB,SB,AR,DQ,VS,G,G1,W,AA,QA,SA)
C
C -----CALCULATE CHANGE IN VARIABLES ACROSS CONTACT SURFACE-----
C
                 QB = QA
                 AB = AA
                 SB = SA
                 SA = S(I2(2)-2)
                 CALL CSJUMP(AB,QB,SB,SA,G2,QA,AA)
                 QQA = QA + AA*SA
                 RRA = QA - AA*SA
                 XA=0.D0
C
C -----INTERPOLATE TO NODE(I2(2)-1) AND ASSIGN CORRECTED VALUES-----
C
                 CALL INTERP(RRA,RR(I2(2)-2),QQA,QQ(I2(2)-2),SA,
     C                  S(I2(2)-2),XA,(XA+H),RR(I2(2)-1),
     C                  QQ(I2(2)-1),S(I2(2)-1),A(I2(2)-1),
     C                  Q(I2(2)-1))
       ELSE
```

```
C
C -----EXTRAPOLATE TO LEFT FACE OF SHOCK-----
C
                CALL EXTRAP(RR(I2(1)-1),RR(I2(1)-2),QQ(I2(1)-1),QQ(I2(1)-2),
      C                  S(I2(1)-1),S(I2(1)-2),H,H,RRA,QQA,SA,AA,QA)
C
C -----CALCULATE CHANGE IN VARIABLES ACROSS SHOCK-----
C
                     CALL SKJUMP(AA,QA,SA,AR,DQ,VS,G,G1,W,AB,QB,SB)
C
C -----CALCULATE CHANGE IN VARIABLES ACROSS CONTACT SURFACE-----
C
                     QA = QB
                     AA = AB
                     SA = SB
                     SB = S(I2(2)+1)
                     CALL CSJUMP(AA,QA,SA,SB,G2,QB,AB)
                     QQB = QB + AB*SB
                     RRB = QB - AB*SB
                     XB=0.D0
C
C -----INTERPOLATE TO NODE(I2(2)) AND ASSIGN CORRECTED VALUES-----
C
                     CALL INTERP(RRB,RR(I2(2)+1),QQB,QQ(I2(2)+1),SB,
      C                     S(I2(2)+1),XB,(XB+H),RR(I2(2)),
      C                     QQ(I2(2)),S(I2(2)),A(I2(2)),Q(I2(2)))
        END IF
        RETURN
        END
C
        SUBROUTINE BONDRY(Q1,Q2,QB,A1,A2,QQ1,QQ2,RR1,RR2,S1,S2,H,EE,
      C                 DELT,BNDRY,BDPRS,BDPR,BDDR,BDTR,J,
      C                 NEWQQ1,NEWRR1,NEWS1,G,G1,G2,HALT,BDRY,SK)
C
C       ******************************************************
C       *                                                    *
C       *            BOUNDARY COUNDITION SUBROUTINE          *
C       *                                                    *
C       ******************************************************
C
C
C     ------------ VARIABLE  DEFINITIONS ------------
C
C     *BD - VARIABLE AT PHANTOM NODE
C     *1 - VARIABLE AT BOUNDARY NODE
C     *2 - VARIABLE AT FIRST NODE INSIDE BOUNDARY
C     D2 - DENSITY RATIO AT FIRST NODE INSIDE BOUNDARY
C     P2 - PRESSURE RATIO AT FIRST NODE INSIDE BOUNDARY
C     QB - INITIAL VELOCITY AT AN OPEN BOUNDARY
C     T2 - TEMPERATURE RATIO AT FIRST NODE INSIDE BOUNDARY
C
        DIMENSION AINT(3),Z(3),QPRIM(3),APRIM(3),INTEG(3),AAVG(3),
      C         QQBD(6),RRBD(6),SBD(6),QBD(6),ABD(6)
        INTEGER BNDRY,BDPRS,J,HALT,SK,BDRY,K,L,M,T
        DOUBLE PRECISION RRBD,QQBD,SBD,QBD,ABD,BDPR,BDDR,BDTR,DELQQH,
      C                 DELQQL,DELRRH,DELSH,DELSL,DELQH,DELAH,DELQL,
      C                 DELAL,H,EE,DELT,QQINT,RRINT,SINT,QPRIM,APRIM,
      C                 AINT,Z,SAVG,AAVG,Q1,Q2,A2,A1,RR1,QQ1,S1,QQ2,RR2,
      C                 S2,DLTAQQ,DLTARR,DLTAS,INTEG,QQSTEP,RRSTEP,D2,
      C                 SSTEP,NEWQQ1,NEWRR1,NEWS1,G,G1,G2,DELRRL,T2,P2,
      C                 AR,DQ,VS,W,QB,NEWTR,NEWDR,NEWPR,EE1,EE2,QRR,QQQ
        COMMON AR,DQ,VS,W
C
C ----- DETERMINE CURRENT PRESSURE AT NODE JUST PRIOR TO BOUNDARY -----
C
        T2 = ((QQ2-RR2)**2)/(4.D00*(S2**2))
        D2 = ((1.D00/T2)*DEXP(G*(1.D00-G)*(S2-G2)))**(-G1)
        P2 = T2 * D2
C
```

```
C ----- DETERMINE IF LEFT OR RIGHT BOUNDARY -----
C
        IF(BDRY.EQ.3) THEN
                K = 1
                L = 2
                M = 3
        ELSE
                K = 6
                L = 5
                M = 4
        END IF
        IF((J.EQ.1).AND.(BNDRY.EQ.0).AND.(BDPRS.EQ.0)) THEN
        QBD(K) = 0.0D0
        END IF
        QQBD(L) = QQ1
        QQBD(M) = QQ2
        RRBD(L) = RR1
        RRBD(M) = RR2
        SBD(L) = S1
        SBD(M) = S2
        QBD(L) = Q1
        QBD(M) = Q2
        ABD(L) = A1
        ABD(M) = A2
C
C ----- DETERMINE IF BOUNDARY IS OPEN OR CLOSED, AND SET PROPER -----
C ----- VALUES AT PHANTOM NODE "LBD". BNDRY = 1(CLOSED),0(OPEN) -----
C
      5 IF (BNDRY.EQ.1) THEN
                RRBD(K) = -QQBD(M)
                QQBD(K) = DABS(RRBD(M))
                  SBD(K) = SBD(M)
                  QBD(K) = -QBD(M)
                  ABD(K) = ABD(M)
                  GO TO 10
        ELSE IF (BDPRS.EQ.1) THEN
                IF (J.EQ.1) THEN
                        RRBD(K) = RRBD(L)
                        QQBD(K) = QQBD(L)
                          SBD(K) = SBD(L)
                          ABD(K) = ABD(L)
                          QBD(K) = QB
                END IF
                GO TO 10
        ELSE IF ((BDPR-P2).LE.0.1D00) THEN
C          IF(SK.EQ.3) GO TO 35
                SBD(K) = SBD(L)
                BDDR = ((1.0D0/BDPR)*(DEXP((SBD(K)-G2)*(-(G/G1)))))
     C                  **(-(1.0D0/G))
                BDTR = ((BDDR)**(1.0D0/G1))*DEXP(G*(1.0D0-G)*(SBD(K)-G2))
                RRBD(K) = QBD(K)-(DSQRT(BDTR))*SBD(K)
                QQBD(K) = QBD(K)+(DSQRT(BDTR))*SBD(K)
                ABD(K) = (QQBD(K) - RRBD(K)) / (2.D00*SBD(K))
                  RRBD(L) = RRBD(K)
                  QQBD(L) = QQBD(K)
                    SBD(L) = SBD(K)
                    QBD(L) = QBD(K)
                    ABD(L) = ABD(K)
                GO TO 10
        ELSE
              PRINT * ,'THE OPEN BOUNDARY HAS A PRESSURE HELD CONSTANT '
              PRINT * ,'THAT IS HIGHER THAN THE PRESSURE INSIDE THE TUBE'
              PRINT * ,'ADDITIONAL CODE IS NEEDED TO PRECEDE'
              HALT = 1
              GO TO 20
        END IF
C
C ----- CALCULATE THE JUMP TO THE NEXT TIME STEP USING CHARACTERISTICS--
C
```

```
C
C --- CHECK FOR SHOCK LEAVING TUBE OR WITHIN H OF BOUNDARY AND PICK --
C --- CHOSE APPROPRIATE ALGORITHM TO ADVANCE BOUNDARY TO NEXT TIME ---
C
   10 IF(SK.EQ.2) THEN
            IF(BDRY.EQ.2) THEN
                  CALL SKJUMP(ABD(L+1),QBD(L+1),SBD(L+1),AR,DQ,VS,G,G1,H,
     C                   ABD(L),QBD(L),SBD(L))
                  QQBD(L) = QBD(L) + ABD(L)*SBD(L)
                  RRBD(L) = QBD(L) - ABD(L)*SBD(L)
                  GO TO 35
            ELSE IF(BDRY.EQ.3) THEN
                  CALL SKJUMP(ABD(L-1),QBD(L-1),SBD(L-1),AR,DQ,VS,G,G1,H,
     C                   ABD(L),QBD(L),SBD(L))
                  QQBD(L) = QBD(L) + ABD(L)*SBD(L)
                  RRBD(L) = QBD(L) - ABD(L)*SBD(L)
                  GO TO 35
            END IF
      END IF
      DELQQL = QQBD(L) - QQBD(L-1)
      DELQQH = QQBD(L+1) - QQBD(L)
      DELRRH = RRBD(L+1) - RRBD(L)
      DELRRL = RRBD(L) - RRBD(L-1)
        DELSH = SBD(L+1) - SBD(L)
        DELSL = SBD(L) - SBD(L-1)
        DELQH = QBD(L+1) - QBD(L)
        DELQL = QBD(L) - QBD(L-1)
        DELAH = ABD(L+1) - ABD(L)
        DELAL = ABD(L) - ABD(L-1)
      IF((BNDRY.EQ.0).AND.(SK.EQ.1)) THEN
              IF (BDRY.EQ.2) THEN
              CALL COND3(QBD(L),QBD(L+1),ABD(L),ABD(L+1),RRBD(L),QQBD(L),
     C                SBD(L),DELQQH,DELRRH,DELSH,DELQH,DELAH,DELT,H,EE,
     C                QQINT,RRINT,SINT,QPRIM,APRIM,AINT)
              ELSE
              CALL COND2(QBD(L),QBD(L-1),ABD(L),ABD(L-1),RRBD(L),QQBD(L),
     C                SBD(L),DELQQL,DELRRL,DELSL,DELQL,DELAL,DELT,H,EE,
     C                QQINT,RRINT,SINT,QPRIM,APRIM,AINT)
              END IF
      ELSE IF((BNDRY.EQ.1).AND.(SK.EQ.1)) THEN
              QQSTEP = 0.D00
              RRSTEP = 0.D00
                SSTEP = 0.D00
                GO TO 30
C
C ----- USE CONDITION 1 ALGORITHM TO CALCULATE RRINT,QQINT,SINT -----
C
      ELSE
      CALL COND1(QBD(L),QBD(L+1),ABD(L),ABD(L+1),RRBD(L),QQBD(L),
     C           SBD(L),DELQQL,DELRRH,DELSH,DELSL,
     C           DELQH,DELAH,DELQL,DELAL,H,EE,DELT,QQINT,RRINT,SINT,
     C           QPRIM,APRIM,AINT,QBD(L-1),ABD(L-1))
      END IF
C
C    --------  CALCULATE DLTA QQ, DLTA RR & DLTA S
C
         DLTAQQ=QQINT-QQBD(L)
         DLTARR=RRINT-RRBD(L)
         DLTAS=SINT-SBD(L)
C
C      ------  CALCULATE  Z(K)'S -----------
C
C
         AAVG(1)=(AINT(1)+ABD(L))/2.0D00
         AAVG(3)=(AINT(3)+ABD(L))/2.0D00
         AAVG(2)=0.0D00
         SAVG = (SINT+SBD(L))/2.D00
         Z(1)=-(1.0D00/G2)*AAVG(1)*(SAVG-G2)*(QPRIM(1)-G2*APRIM(1))
         Z(3)=(1.0D00/G2)*AAVG(3)*(SAVG-G2)*(QPRIM(3)+G2*APRIM(3))
```

```
         Z(2)=0.0D00
C
C       ----- INTEGRATE THE Z(K)'S ------------
C
         INTEG(2)=0.0D00
         INTEG(1)=Z(1)*DELT
         INTEG(3)=Z(3)*DELT
C
C       -------- SOLVE THE EQUATION -----------
C
         QQSTEP=DLTAQQ+INTEG(1)
         RRSTEP=DLTARR+INTEG(3)
         SSTEP=DLTAS+INTEG(2)
C
C       -------- STORE THE SOLUTION -----------
C
   30   NEWQQ1=QQBD(L)+QQSTEP
         NEWRR1=RRBD(L)+RRSTEP
         NEWS1=SBD(L)+SSTEP
C
C ----- UPDATE PHANTOM NODES FOR OPEN BOUNDARY CONDITION -----
C
   35   IF ((BNDRY.EQ.0).AND.(BDPRS.EQ.1))THEN
                 RRBD(K) = RRBD(L)
                 QQBD(K) = QQBD(L)
                 SBD(K) = SBD(L)
                 ABD(K) = ABD(L)
                 QBD(K) = QBD(L)
         END IF
         IF ((BNDRY.EQ.0).AND.(BDPRS.EQ.0)) THEN
                 IF (SK.EQ.2) THEN
                 QBD(K) = (QQBD(L)+RRBD(L)) / 2.D00
                 ELSE
                 NEWTR= ((NEWQQ1 - NEWRR1)**2)/(4.D00*((NEWS1)**2))
                 NEWDR= ((1.D00/NEWTR)*DEXP(G*(1.D00-G)*(NEWS1-G2)))
     C                 **(-G1)
                 NEWPR= NEWTR*NEWDR
                 EE1 = DABS(BDPR-NEWPR)
                 EE2 = DABS(SBD(L)-NEWS1)
                 IF ((EE1.GT.0.1D-5).OR.(EE2.GT.0.1D-5)) THEN
                      QRR= NEWRR1 + (DSQRT(BDTR))*SBD(L)
                      QQQ= NEWQQ1 - (DSQRT(BDTR))*SBD(L)
                      QBD(K) = (QRR+QQQ)/2.DDD
                      GO TO 5
                 END IF
                 END IF
            END IF
   20 QQ1 = QQBD(L)
       RR1 = RRBD(L)
       S1 = SBD(L)
       Q1 = QBD(L)
       A1 = ABD(L)
       RETURN
       END
C
       SUBROUTINE SRFLCT(QQA,RRA,SA,SIGMA,VS,DELT,LWPRES,RRB,QQB,SB,QB,
     C                   AB,G,G1,G2)
C
C      ******************************************************
C      *                                                    *
C      *    SHOCK REFLECTION AT SOLID BOUNDARY SUBROUTINE   *
C      *                                                    *
C      ******************************************************
C
C
C     ------------ VARIABLE  DEFINITIONS ------------
C
C     DELTEX - THE EXCESS TIME IN A TIME STEP WHEN THE SHOCK IS
C              EXACTLY AT THE SOLID WALL
```

```
C     DELTWL - THE TIME FOR THE SHOCK TO REACH THE WALL
C
          DIMENSION SIGMA(4,2)
          INTEGER LWPRES
          DOUBLE PRECISION QA,QB,AA,AB,SA,SB,QQA,QQB,RRA,RRB,SIGMA,
        C                DQ,W,AR,PR,DR,EREIMN,SA1,SA2,VS,DELT,DELTWL,
        C                DELTEX,G,G1,G2
C
C ----- CALCULATE THE TIME FOR SHOCK TO REACH WALL, AND EXCESS TIME ----
C ----- IN THIS TIME STEP -----
C
          IF (LWPRES.EQ.2) THEN
          DELTWL = (1.D00-SIGMA(1,1))/VS
          DELTEX = DELT - DELTWL
C
C ----- CALCULATE VELOCITY GRADIENT OVER SHOCK AS IT REFLECTS -----
C
          QA = (QQA+RRA)/2.D00
          QB = 0.D00
          AA = (QQA-RRA)/(2.D00*SA)
          DQ = (QB-QA)/AA
C
C ----- CALCULATE W -----
C
          W = DSQRT(((DQ**2)*0.36D00)+1.D00) - (DQ*0.6D00)
C
C ----- FOR SHOCK AT LEFT BOUNDRY DO THE SAME -----
C
          ELSE
          QQB=QQA
          RRB=RRA
           SB=SA
          DELTWL = (DABS(SIGMA(1,1))-0.D00)/DABS(VS)
          DELTEX = DELT - DELTWL
C
C ----- CALCULATE VELOCITY GRADIENT OVER SHOCK AS IT REFLECTS -----
C
          QB = (QQB+RRB)/2.D00
          QA = 0.D00
          AB = (QQB-RRB)/(2.D00*SB)
          DQ = (QA-QB)/AB
C
C ----- CALCULATE EXACT REIMAN VARIABLE JUMP FROM DQ -----
C
          W = DSQRT(((DQ**2)*0.36D00)+1.D00) + (DQ*0.6D00)
          END IF
C
C ----- CALCULATE AR,PR,DR OVER SHOCK -----
C
          AR=DSQRT(2.D00*(G-1.D00)*(1.D00+((G-1.D00)*W*W/2.D00))*
        C          (G*G2*W*W-1.D00))/((G+1.D00)*W)
          PR=(2.D00*G/(G+1.D00))*W*W-((G-1.D00)/(G+1.D00))
          DR=((G-1.D00)*W*W+2.D00)/((G+1.D00)*W*W)
          SA1 = (G1/G)*DLOG((2.D00*G*(W**2)-G+1.D00)/(G+1.D00))
          SA2 = G1*DLOG(((G-1.D00)*(W**2)+2)/((G+1.D00)*(W**2)))
C
C ----- CALCULATE ENTROPY AND SPEED OF SOUND BEHIND SHOCK -----
C
          IF (LWPRES.EQ.2) THEN
          SB = SA - SA1 - SA2
          AB = AA*AR
C
C ----- CORRECT REIMAN VARIABLES AT BOUNDARY -----
C
          RRB = QB - AB*SB
          QQB = QB + AB*SB
C
C ----- CALCULATE NEW SHOCK SPEED -----
C
```

```
      VS = ((RRA+QQA)*0.5D00) - (W*AA)
C
C ----- CALCULATE POSITION OF REFLECTED SHOCK AT END OF THIS TIME STEP--
C
      SIGMA(1,2) = VS*DELTEX + 1.D00
      LWPRES = 3
C
C ----- SHOCK REFLECTING AT LEFT BOUNDARY -----
C
      ELSE
      SA = SB - SA1 - SA2
      AA = AB*AR
C
C ----- CORRECT REIMAN VARIABLES AT BOUNDARY -----
C
      RRA = QA - AA*SA
      QQA = QA + AA*SA
C
C ----- CALCULATE NEW SHOCK SPEED -----
C
      VS = ((RRB+QQB)*0.5D00) + (W*AB)
C
C ----- CALCULATE POSITION OF REFLECTED SHOCK AT END OF THIS TIME STEP--
C
      SIGMA(1,2) = VS*DELTEX
      LWPRES = 2
      RRB=RRA
      QQB=QQA
        SB=SA
        QB=QA
        AB=AA
      END IF
      RETURN
      END
C
      SUBROUTINE BBDRY(RR1,QQ1,S1,RR2,QQ2,S2,A2,Q2)
C
C     ***************************************************
C     *                                                 *
C     *          NODES NEAR BOUNDARY SUBROUTINE         *
C     *                                                 *
C     ***************************************************
C
      DOUBLE PRECISION RR1,QQ1,S1,RR2,QQ2,S2,A2,Q2
      RR2 = RR1
      QQ2 = QQ1
        S2 = S1
        A2 = (QQ2 - RR2)/(2.D00*S2)
        Q2 = (QQ2 + RR2)/2.D00
      RETURN
      END
C
C
      SUBROUTINE BORDER(JSTOP)
C
C   ********************************************
C   *                                          *
C   *    PLOTTING AREA SETUP ROUTINE           *
C   *                                          *
C   ********************************************
C
      INTEGER I
      REAL XORG(4)/1.75,4.65,1.75,4.65/
      REAL YORG(4)/2.75,2.75,5.80,5.80/
      REAL YMAX(4)/5.5,5.5,1.5,6.20/
      REAL YMIN(4)/0.5,0.5,-0.5,4.90/
      DO 70 I=1,4
           CALL PHYSOR(XORG(I),YORG(I))
           CALL NOBRDR
```

```
           CALL AREA2D(2.4,2.4)
           CALL FRAME
           CALL GRAF(0.,'SCALE',1.0,YMIN(I),'SCALE',YMAX(I))
           CALL ENDGR(0)
   70  CONTINUE
         CALL PHYSOR(1.25,2.25)
         CALL NOBRDR
         CALL AREA2D(6.00,6.50)
         CALL HEADIN('SHOCK TUBE RESULTS$',100,1.7,4)
         CALL HEADIN('FIRST ORDER      N =101$',100,1.2,4)
         CALL HEADIN('DENSITY RATIO = 5.0   TEMP RATIO = 1$',100,1.0,4)
         CALL HEADIN('PRESSURE RATIO = 5.0$',100,1.0,4)
         CALL GRAF(0.,'SCALE',1.,0.,'SCALE',JSTOP)
         CALL ENDGR(0)
         RETURN
         END
C
C
         SUBROUTINE PLOT(J,JSTOP,N,QQ,RR,S,H,XARRAY,
        #PARRAY,DARRAY,QARRAY,SARRAY,G,G1,G2)
C
C     *********************************************
C     *                                           *
C     *        GRAPHICAL  PLOTTING  ROUTINE       *
C     *                                           *
C     *********************************************
C
         INTEGER I,N,J,JSTOP,KNT(4)/1,4,6,9/
         DIMENSION QQ(N),RR(N),S(N),XARRAY(N),QARRAY(N),PARRAY(N),
        C          DARRAY(N),SARRAY(N)
         DOUBLE PRECISION QQ,RR,S,H,G,G1,G2
         REAL XARRAY,QARRAY,PARRAY,DARRAY
         REAL SARRAY,TEMP,HR,G1R,GR,G2R
         REAL XORG(4)/1.75,4.65,1.75,4.65/
         REAL YORG(4)/2.75,2.75,5.80,5.80/
         REAL YMAX(4)/5.5,5.5,1.0,6.20/
         REAL YMIN(4)/0.5,0.5,-1.0,4.90/
         CHARACTER*4 IYNAM
         DIMENSION IYNAM(13)
         DATA IYNAM/'PRES','SURE','$   ','DENS',
        #'ITY$','VELO','CITY','$   ','MODI','FIED',' ENT','ROPY','$   '/
C
C --- CONVERT DOUBLE PRECISION TO SINGLE PRECISION ---
C
         GR=SNGL(G)
         G1R=SNGL(G1)
         G2R=SNGL(G2)
         HR=SNGL(H)
         DO 80 I=1,N
         QARRAY(I)=(SNGL(QQ(I)+RR(I))/2.0)
         TEMP=SNGL(QQ(I)-RR(I))*SNGL(QQ(I)-RR(I))/(4.0*SNGL(S(I)*S(I)))
         DARRAY(I)=((1.0/TEMP)*EXP(GR*(1.0-GR)*(SNGL(S(I))-G2R)))**(-G1R)
         SARRAY(I)=SNGL(S(I))
         PARRAY(I)=TEMP*DARRAY(I)
   80  CONTINUE
         DO 83 I=1,4
              CALL PHYSOR(XORG(I),YORG(I))
              CALL AREA2D(2.4,2.4)
              CALL XNAME('X',1)
              CALL YNAME(IYNAM(KNT(I)),100)
              CALL GRAF(0.,'SCALE',1.0,YMIN(I),'SCALE',YMAX(I))
              IF (I.EQ.1) CALL CURVE(XARRAY,PARRAY,N,0)
              IF (I.EQ.2) CALL CURVE(XARRAY,DARRAY,N,0)
              IF (I.EQ.3) CALL CURVE(XARRAY,QARRAY,N,0)
              IF (I.EQ.4) CALL CURVE(XARRAY,SARRAY,N,0)
              CALL ENDGR(0)
     83 CONTINUE
         RETURN
         END
```

```
C
      SUBROUTINE EXACT(N,XINIT,T,VHEAD,VTAIL,VCDE,VSE,DLCD,DLSH,QQ,RR,S,
     #H,XARRAY,DARRAY,G,G1,G2,DRI)
C
C    ************************************************
C    *                                              *
C    *     EXACT SOLUTION COMPARISON ROUTINE        *
C    *                                              *
C    ************************************************
C
      INTEGER N
      DIMENSION QQ(N),RR(N),S(N),XEXACT(6),YEXACT(6),XARRAY(N),
     C          DARRAY(N)
      DOUBLE PRECISION QQ,RR,S,H,G,G1,G2,T,DRI
      REAL XEXACT,YEXACT,XINIT,VHEAD,VTAIL,VCDE
      REAL VSE,DLCD,DLSH,TEMP,XARRAY,DARRAY
C
      XEXACT(1)=0.0
      XEXACT(2)=XINIT+T*VHEAD
      XEXACT(3)=XINIT+T*VTAIL
      XEXACT(4)=XINIT+T*VCDE
      XEXACT(5)=XINIT+T*VSE
      XEXACT(6)=1.0
      YEXACT(1)=SNGL(DRI)
      YEXACT(2)=YEXACT(1)
      YEXACT(3)=DLCD
      YEXACT(4)=YEXACT(3)
      YEXACT(5)=DLSH
      YEXACT(6)=1.0
C
      DO 90 I=1,N
      QQ(I)=SNGL(QQ(I))
      RR(I)=SNGL(RR(I))
      S(I)=SNGL(S(I))
      TEMP=(QQ(I)-RR(I))*(QQ(I)-RR(I))/(4.0*S(I)*S(I))
      DARRAY(I)=((1.0/TEMP)*EXP(G*(1.0-G)*(S(I)-G2)))**(-G1)
   90 CONTINUE
      CALL PHYSOR(2.65,2.25)
      CALL NOBRDR
      CALL AREA2D(6.75,4.0)
      CALL XNAME('X',1)
      CALL YNAME('DENSITY',7)
      CALL HEADIN('DENSITY DISTRIBUTION$',100,1.3,4)
      CALL HEADIN('FIRST ORDER       N =101$',100,1.0,4)
      CALL HEADIN('DENSITY RATIO = 5    TEMP RATIO = 1$',100,1.0,4)
      CALL HEADIN('PRESSURE RATIO = 5$',100,1.0,4)
      CALL LINES('EULER-1 SOLUTION$',IPKRAY,2)
      CALL LINES('EXACT SOLUTION$',IPKRAY,1)
      CALL FRAME
      CALL GRAF(0.0,'SCALE',1.0,0.0,'SCALE',6.0)
      CALL MARKER(0)
      CALL CURVE(XEXACT,YEXACT,6,-1)
      CALL LEGLIN
      CALL CURVE(XARRAY,DARRAY,N,0)
      CALL LEGEND(IPKRAY,2,4.25,2.75)
      CALL ENDGR(0)
      RETURN
      END
```

LIST OF REFERENCES

1. McDonnell Aircraft Company Report 83-031, A Natural Formulation for Numerical Solutions of the Euler Equations, by A. Verhoff and P.J. O'Neil, 1983.

2. Salacka, T.F., Review, Implementation and Test of the QAZ1D Computational Method with a View to Wave Rotor Applications, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1985.

3. NASA Contractor Report 3712, An Improved Lambda-Scheme for One-Dimensional Flows, by G. Moretti and M.T.D; Piano, 1983.

4. Zucrow, M.J. and Hoffman, J.D., Volume II Gas Dynamics Multidimensional Flows, Robert E. Krieger Publishing Company, 1985.

5. Lapidus, L. and Pinder, G.F., Numerical Solution of Partial Differential Equations in Science and Engineering, John Wiley & Sons, Inc., 1982.

6. Wright, J.K., Shock Tubes, John Wiley & Sons, Inc., 1961.

7. Zucker, R.D., Fundamentals of Gas Dynamics, Matrix Publishers, Inc., 1977.

8. Mathur, A., Riemann (Fortran Code), Personal Communication, February 1987.

9. Courant, R. and Friedrichs, K.O., Supersonic Flows and Shock Waves, Interscience Publishers, Inc., 1948.

10. Shapiro, A.H., The Dynamics and Thermodynamics of Compressible Fluid Flow Vol. I, John Wiley & Sons, Inc., 1953.

11. Zemansky, M.W., Abbott, M.M., and Van Hess, H.C., Basic Engineering Thermodynamics, McGraw-Hill, Inc., 1966.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	A. Verhoff, Code D341 McDonnell Douglas Corporation Box 516 St. Louis, Missouri 63166-0516	3
4.	L.T. Johnston Box 35 RR 1 Altona, Illinois 61414	3
5.	Office of Research Administration, Code 012 Naval Postgraduate School Monterey, California 93943-5000	1
6.	Chairman, Code 67 Department of Aeronautics Naval Postgraduate School Monterey, California 93943-5000	1
7.	Director, Turbopropulsion Laboratory, Code 67SF Department of Aeronautics Naval Postgraduate School Monterey, California 93943-5000	15
8.	Dr. Gerhard Heiche Naval Air Systems Command, Code 93D Washington, D.C. 20361-0001	1
9.	Mr. George Derderian Naval Air Systems Command, Code 931E Washington, D.C. 20360-0001	1
10.	Dr. F. Hansen Office of Naval Research, Code 12D 800 North Quincy Street Arlington, Virginia 22217-0001	2

11. Professor Ch. Hirsch 1
Vrije Universiteit Brussel
Pleinlaan 2
1050 Brussels
Belgium

12. Mr. P. Tramm 1
Allison Gas Turbine Division
of General Motors
P.O. Box 420
Indianapolis, Indiana 46206-0420

13. Calvin Ball, Small Gas Turbine Engines 1
NASA Lewis Research Center, MS77-6
21000 Brookpark Road
Cleveland, Ohio 44134-1525

14. David Gordon Wilson 1
M.I.T. Mechanical Engineering, Room 3-455
Cambridge, Massachusetts 02139

15. Helmut E. Weber 1
Professor, Department of
Mechanical Engineering
San Diego State University
San Diego, California 92182-0191

16. Robert Taussig 1
Director, Energy Technology
Spectra Technology, Inc.
2755 Northup Way
Bellevue, Washington 98004-1495

17. Dr. Shmuel Eidelman, Research Physicist 1
Science Application International Co.
1710 Goodridge Dr. Mail Stop G-8-1
McLean, Virginia 22306

18. LCDR T.F. Salacka 1
VQ-4
NAS Patuxent River, Maryland 20670

Thesis
J663 Johnston
c.1 Further development of
a one-dimensional unsteady
Euler code for wave rotor
applications.